MB-6885

ベーシックマスターJr.

取扱説明書

HITACHI

特　長

■BASIC，機械語，および別売のアセンブラーの3種類の言語が使用可能です。

■テキストモード（通常の文字・図形モード）1ページの外に，鮮明なグラフィック表示が可能な高解像度グラフィックモード（256×192ドット）を2ページ，計3ページの表示モードを標準実装しています。

■プリンターインターフェイスを内蔵，さらにミニフロッピーディスク等の周辺機器を駆動する拡張ポートを内蔵しています。

■最大9桁（浮動小数点）の精度の高い計算が可能。

■三角関数，文字取扱い関数をはじめとする豊富な関数群を内蔵。

■画面上でプログラムの作成・編集ができます。編集は1文字単位で，かつ内蔵のプログラム編集コマンドの活用により極めて容易に行えます。

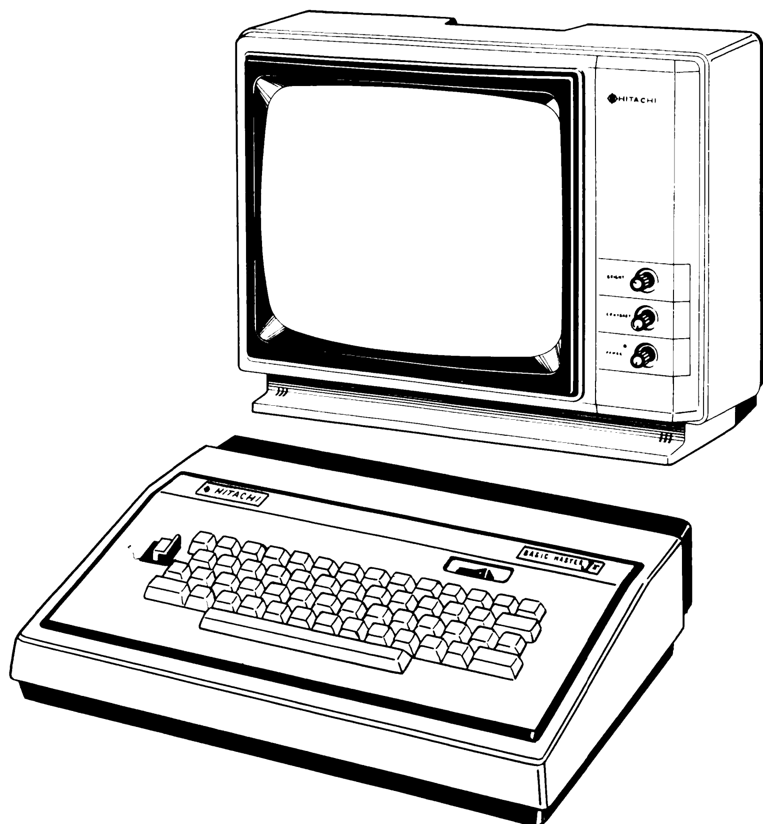
■ステップスカルプチャータイプのキーボードを採用，操作性が一段と良くなりました。

■内蔵RAMは16KB，別売の64Kビットダイナミック RAMを拡張することにより最大63.5KBまで拡張できます。

■RAMの拡張はレジスター切換方式によって，次のように拡張でき広いユーザーエリアを確保できます。

使用状態	BASIC使用時	モニター使用時	オールRAM化時
RAM	44KB	56KB	63.5KB

■別売のカラーアダプターを接続することによりカラー表示ができます。



使用上のご注意

ご使用にあたっては，必ず以下のご注意をお守りください。お守りいただけない場合には品質の保証をいたしかねることがあります。

■電源コードは大切に

電源コードをセットの下に敷いたりして，傷をつけないようにご注意ください。電源コードに傷がついたまま使用することは危険です。また，電源コードを引っ張らないで必ずプラグを持って抜いてください。

■電源電圧のご注意

電源電圧は，90V～110Vの範囲内でお使いください。この範囲外でご使用になりますと，正常な動作をしなくなる場合があります。

■使用温度のご注意

本セットは，0℃～35℃の範囲でお使いください。この範囲外でご使用になりますと，正常な動作をしなくなる場合があります。

■使用湿度のご注意

湿度は，20%～80%の範囲内でお使いください。この範囲外でお使いになりますと，正常な動作をしなくなる場合があります。

■風通しの良い所で

本セットは温度上昇を防ぐため，ケースに通風孔をあけてありますので，通風孔をふさいだり，風通しの悪い場所でのご使用はさけてください。

■ほこりの少ない場所で

ほこりの多い場所での保管およびご使用はさけてください。正常な動作をしなくなる場合があります。

■衝撃・振動は故障の原因

セットは，精密な電子部品でできていますので，衝撃を加えたり，衝撃振動の加わる場所での保管およびご使用はさけてください。

■薬品は禁物

薬品の雰囲気中や薬品に触れる場所での保管およびご使用はさけてください。

■水は禁物

セットの内部に水や液状のものが入った場合，そのまま使用しますと，故障の原因となると同時に危険でもあります。すぐに電源コードのプラグをコンセントから抜き，販売店にご連絡ください。

■異物を入れると危険

セットのすきまなどから金属類や燃えやすいものを差し込んだり，落したりすると感電や火事の原因となります。

■ケースは取りつけた状態で

ケースを取りはずした状態での保管およびご使用は，故障や感電の原因になりますのでおやめください。またセット内部には電圧の高い部分があります。危険ですのでセット内部には触れないでください。

■セットの上には物をおかないで

セットの上に重い物や，カセットテープなどを置いた状態での保管およびご使用はおやめください。

■コネクターには触れないで

プリンター接続コネクターや拡張ポートのコネクターカバーは，周辺機器を接続する時以外はずさない様にしてください。また，コネクターの端子に手を触れないでください。触れますと故障の原因となります。

■電源の再投入時のご注意

電源スイッチを切り，直後に再投入しますと，3頁の画面が出ないことがあります。これはセットの内部状態が初期状態に戻らないうちに再起動をかけられた結果によるものであり，故障ではありません。このようなときは一度電源スイッチを切り，5秒以上たってから再び電源スイッチを入れるようにしてください。

■ラジオ・テレビに雑音が入ったときは

ラジオやテレビなどの近くで使用しますと，ラジオやテレビに雑音が入ることがあります。また，強い磁界や雑音を発生する装置などが近くにありますと，逆にセットに雑音が入ってくることがあります。このような場合は離してご使用ください。

■表面のお取り扱い

セットの汚れは，やわらかい布でからぶきをしてください。汚れがひどい場合は，中性洗剤を薄め，布に浸して固く絞り，汚れをふきとったのち，乾いた布で仕上げをしてください。ベンジン・シンナーなど揮発性のものや，化学ぞうきんなどでふいたり，殺虫剤をかけたりしますと，セット表面が変質することがあります。また，ゴムやビニール製品を長時間接触させておきますと，シミがつくことがありますので，ご注意ください。

■長時間ご使用にならないとき

ご旅行などで長い間ご使用にならないときは，必ず電源コードのプラグをコンセントから抜いておいてください。

■故障や異常のときは

万一故障や異常(臭い，過熱など)にお気づきのと

きは，すぐ電源スイッチを切り電源プラグをコンセントから抜いてお買い求めの販売店に修理をご依頼ください。また転居その他でお困りの場合は，同梱包の日立家電品ご相談窓口一覧表をご覧の上，サービス窓口にご相談ください。

■カセットテープレコーダーとの接続時のご注意

本セットのカセット録音再生特性は，日立カセットテープレコーダーＴＲＱ－237およびＴＲＱ－359を基本に設定してありますので，他のカセットテープレコーダーを使用した場合正常な動作をしない場合があります。このようなときはお買い求めの販売店にご相談ください。

■ディスプレイ画面に手を触れないで

ディスプレイやテレビ使用時に画面に手を触れますと，帯電する場合がありますので，なるべくディスプレイ画面には手を触れることのないようご注意ください。

■セット使用中はディスプレイの電源を切らないで

セット使用中に，ディスプレイの電源をお切りになりますと，正常に動作しない場合がありますので，セット使用中は，ディスプレイの電源をお切りにならないでください。

■他装置への接続は

本セットの拡張ポートや，プリンター端子など本体から他の装置への接続は，必ず指定された装置をお使いください。それ以外の装置の接続による場合は，品質の保証をいたしかねることがあります。また指定された装置をお使いになる場合はその装置の取扱説明書に従ってください。

■拡張するときは

インターフェイスやメモリーなどの拡張はすべて日立ベーシックマスター取扱店にてサービスマンが行います。万一，上記以外の方法により拡張を行われました場合は，品質の保証をいたしかねる場合があります。また，ベーシックマスタージュニアのハードウェア，ソフトウェアを改造したものについては，弊社は一切の責任を負いません。

■プリンターとの接続ご注意

本セットのプリンターインターフェイスは，日立ドットインパクトプリンターMP-1041およびMP-1045を基本に設定しておりますので，他のプリンターをお使いの場合は正常な動作をしない場合があります。

■はげしい雷のときは

はげしい雷のときは，落雷などによる事故防止上電源スイッチを切り，電源コードのプラグをコンセントから抜いていただくと万全です。

目　次

BASIC編

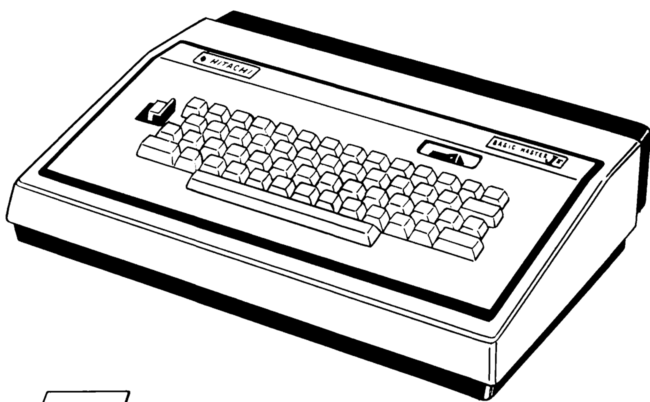
モニター編

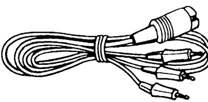
その他

1.取り扱い方法

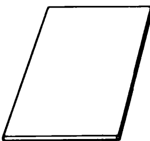
1.1 部品の確認

下記の部品が全て入っていることをご確認ください。

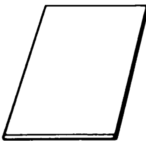




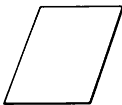
カセットケーブル



取扱説明書



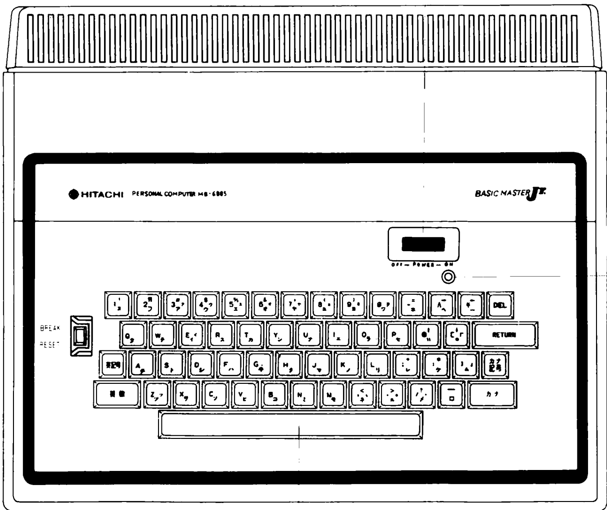
使用の手引き



保証書
日立家電品ご相談窓口一覧表

1.2 各部の名称とお取り扱い

電源スイッチ　白い点のある方を押すと電源が入りランプが点灯します。

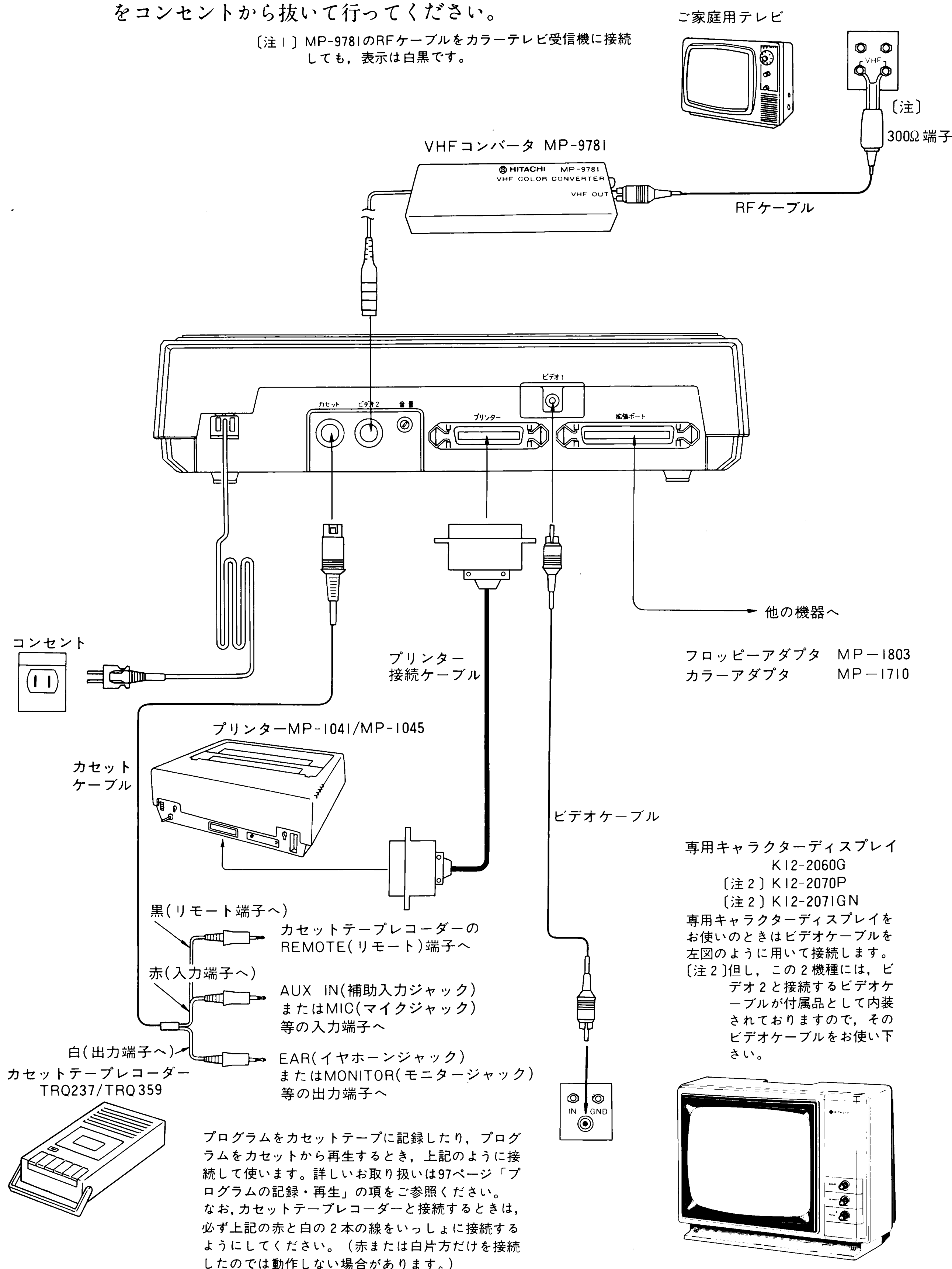


ランプ(電源ON表示)

キーボード
詳しいお取り扱いは,
本文をご参照ください。

1.3 接続のしかた

本セットと他機器との接続を行うときは必ず電源スイッチを切り，電源コードのプラグをコンセントから抜いて行ってください。

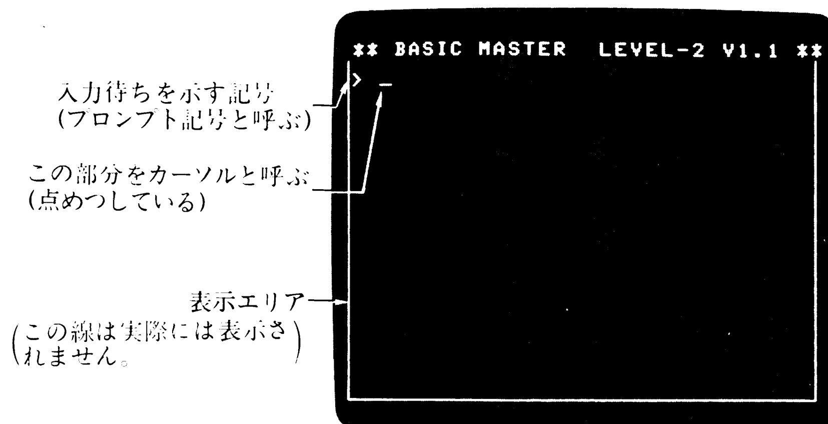


2.操作のしかた

1．接続のしかたに従って，ベーシックマスタージュニア本体とキャラクターディスプレイ，またはＶＨＦコンバータと御家庭用のテレビを接続します。

2．キャラクターディスプレイまたはテレビとベーシックマスタージュニアの電源コードプラグをコンセントに差し込みます。

3．キャラクターディスプレイまたはテレビとベーシックマスタージュニアの電源スイッチをＯＮにしてください。ベーシックマスタージュニアのランプが点灯し，キャラクターディスプレイまたはテレビの画面に(以下単に画面と呼びます。)次に表示がでてきます。

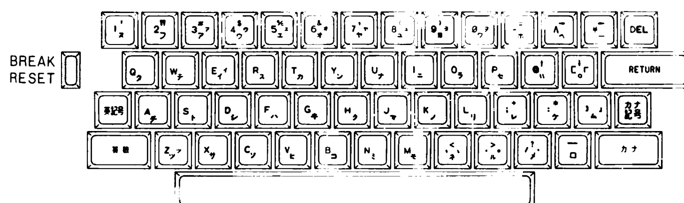


3．これでもうＢＡＳＩＣが〝走ります〟

4．ＢＡＳＩＣによるプログラムをつくり，実行させるにはＢＡＳＩＣの〝文法〟を知っていることが必要です。ＭＢ－6885のＢＡＳＩＣの用語，文法をはじめに確実に理解してください。ＢＡＳＩＣについては別項で説明します。

ヒント

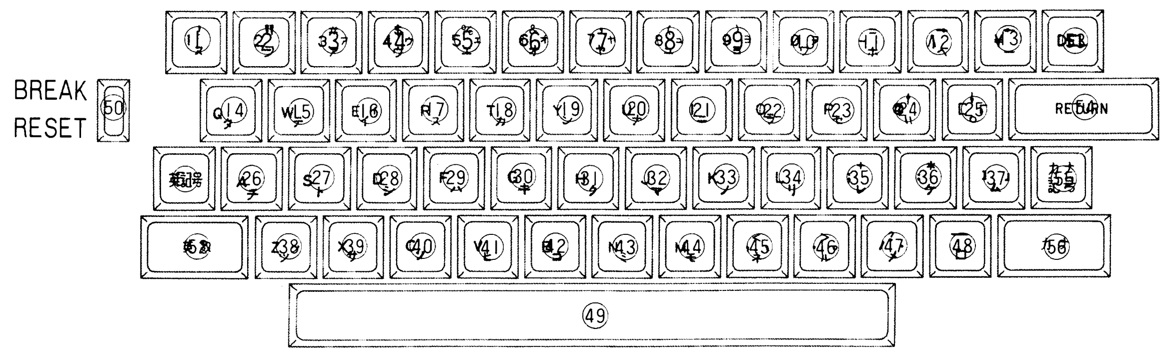
リセット：カナ記号 ＋ BREAK RESET によっても上図の表示が現われます。(電源を入れて上図のような表示がでないときは，この方法により，リセットをかけてください。詳細は6ページ参照)



2.1 キーボードの操作方法　その1

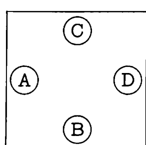
(1) キーボードは下図のように合計56個のキーで構成されています。この内①～㊾までは文字や記号を入力するためのキーで，残りの周辺に配置された㊿～56までのキーが特殊な働きをするキーです。

(2) 各々のキーをどのような組み合わせでおした時プログラムが組まれるかはＢＡＳＩＣ編で説明します。ここではどのような操作でキーをおせば，思った通りの文字や記号が入力されるかを説明します。



(3) ①～㊾迄のキーが文字や記号を入力するためのキーと説明しましたが，たとえば⑥のキーを見てみますと，キーの表面に６，オ，&，ォという４種類の文字や記号が書かれており，単にキーをおしたのではこの４種の内のどれが選択されるかわかりません。この選択を決定するのが51，52，55，56のキーです。この関係を下に示します。

即ち①～㊾のキーを一般的に



とすると，

Ⓐ	52の 英 数 キーをおしてから左の例のキーをおすとⒶの部分(左横部分)に書かれた文字や記号が入力される。(⑥のキーを例にとると６が入力される。)	Ⓑ	56の カ ナ キーをおしてから左の例のキーをおすとⒷの部分(下の部分) に書かれた文字や記号が入力される。(⑥のキーを例にとるとオが入力される。)
Ⓒ	51の 英記号 キーをおしながら左の例のキーをおすとⒸの部分 (上の部分) に書かれた文字や記号が入力される。(⑥のキーを例にとると&が入力される。)	Ⓓ	55の カナ記号 キーをおしながら左の例のキーをおすとⒹの部分 (右横部分) に書かれた文字や記号が入力される。(⑥のキーを例にとるとォが入力される。)

なお，英 数，カ ナ，英記号，カナ記号 の４種のキーのことをシフトキーと呼びます。

(4) (3)の説明でおしてからとなっている所は１度 英 数 なり カ ナ キーをおせば，その後手を離してもその状態が保たれることを意味しています。これに対しおしながらとなっている所は文字通りおしながら操作しないと，その機能が発揮されないことを意味します。
なおここでたとえば①のキーのように右横の部分((3)の説明図のⒹの部分)に何も書かれていないキーでは，この部分を選択するよう カナ記号 キーをおしながら入力しようとしても何も入力されません。(スペースすらも入力されません。)
なお電源スイッチを入れた最初の状態では，自動的に 英 数 キーをおしたと同じ状態が設定されます。

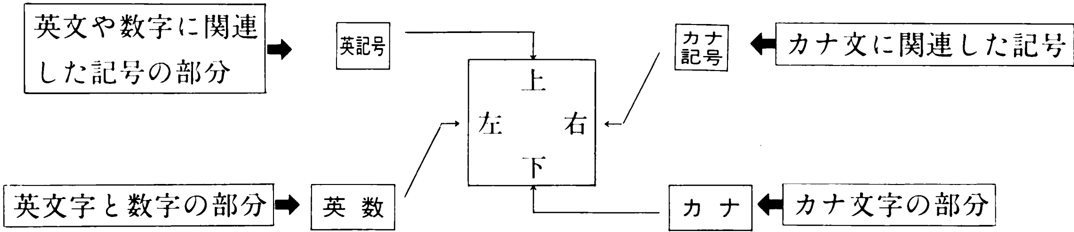
(5) 前の説明でスペースという言葉が出てきましたが，これは1文字分の空白という意味で，このスペースを入力するためには一番手前の長い㊾キーを使います。(このキーはシフトキーがいかなる状態にあろうともいつもスペースを入力します。)

(6) 以上で①～㊾，51，52，55，56のキーの説明は一通り終りですが，これらのキーでの若干の例外事項や注意事項を次にのべます。

キー番号及び部分	表示内容	意味
⑫キー 上部	→	これらの矢印は今までの説明で述べた使用法に従って入力しようとしても入力されません。特殊な用法に用います。36ページ，画面エディターの項で詳しく説明します。
⑬キー 上部	←	
㉔キー 上部	↑	
㉕キー 上部	↓	
②キー 上部	"	"ベーシックマスター"というように用いる引用符(クオーテーションマーク)です。
㉔キー 下部	ヽヽ	ガギグゲゴというように用いるカナ文字に対するだく点です。
⑪キー 左横部	−	−5, 10−5, というように用いるマイナス及び引き算の記号です。
⑬キー 下部	−	ケーキというように用いるカナ文字に対する長音記号です。
㊽キー 上部	−	アナタノナマエハ−−−タロウデスというように用いる破線記号です。
㊻キー 左横部	.	小数点です。英文の終りに用いるピリオドも兼ねます。
㊼キー 右横部	・	アナタノナマエハ・・・タロウデスというように用いる点々を意味します。
㉕キー 下部	°	パピプペポというように用いるカナ文字に対する半だく点です。
㊻キー 右横部	。	…シマシタ。というように用いるカナ文の終りの句点です。
㊺キー 右横部	、	ワタシハ、コンピューターニ…というように用いるカナ文の途中の読点です。
③～⑨ ⑯，㊳キー 右横部	カナ小文字	キャッチボールというように用いるカナ文での促音，拗音です。
㉕，㊲キー 右横部	「及び」	「ハイ」トコタエルト…というように用いるカナ文中のカギカッコです。
㊱キー 上部	＊	かけ算の記号で 4＊5 というように用い，通常の 4×5 と同じ意味です。
㊼キー 左横部	／	わり算の記号で 10/2 というように用い，通常の 10÷2 と同じ意味です。
⑫キー 左横部	^	べき乗算の記号で，5^2 というように用い，通常の5^2と同じ意味です。

上の表で〔でくくった所は，記号が似ていてまちがいやすい所です。
例えば 10−5 という引き算の時にマイナスの記号の代りに⑬や㊽キーの所にある記号を使っても計算はされず，エラーの答が返ってきますから注意が必要です。

(7) (6)でだいぶ例外事項や，注意事項が出てきてめんどうな感じを受けますが，要は



という基本的な関係をおぼえれば，上の表の関係を完全に頭の中に入れなくとも（使っている内に自然と頭の中に入ってくるものですが）ほとんどまちがいなく使うことが出来ます。

(8) 次に残りの㊿，53，54のキーについて説明します。

BREAK RERET ㊿ リセット／ブレークキー

(イ) カナ記号 キーをおしながら BREAK RESET キーをおして離したときＲＥＳＥＴ(リセット)という機能が働きます。この機能はすべての状態を初期状態に戻すことを意味し，一度電源スイッチを切って再び入れたのと同じです。もしＲＥＳＥＴをかける前に何らかのプログラムが入っていてもＲＥＳＥＴをかけた時点でそのプログラムも全て消されてしまいます。従ってうっかりプログラムを消してしまうことのないよう，上記のように2つのキーを同時に操作しない限りＲＥＳＥＴ機能が働かないようになっています。

(ロ) 単に BREAK RESET キーをおすと，ＢＲＥＡＫ(ブレーク)という中断を意味する機能が働き，その時実行中であったプログラムがその時点で中断されます。この時プログラム自体はＲＥＳＥＴの場合と異なり，消されてしまうことはありません。なおＢＲＥＡＫは(イ)で説明した カナ記号 キーを除くシフトキーがどのような状態にあろうとも，単に BREAK RESET キーをおすことにより動作します。(ただしＢＲＥＡＫという機能はプログラムが実行されているときこれを中断するものですから，プログラムが実行されていない場合には BREAK RESET キーをおしても何も反応しません。)

DEL 53 DELETEキー (デリート キー)

まちがった文字や記号を入力してしまった時の訂正用のキーです。このキーをおすと，おすたびに前の１文字を消しながら前に戻って行きます。たとえばABCDEFと入力したいとき，まちがえて A B C G と D の代りに G を入力してしまったら１回 DEL キーをおし，ABCとしてから，またD以降を入力することにより内容が訂正されます。

RETURN 54 RETURNキー (リターン キー)

復帰改行の意味で，1つの文章が終了した時このキーをおすことにより，文字や記号が入力される位置を文章の先頭位置へ戻し(復帰)，次の行へ移る(改行)動作が行なわれます。また，RETURN 後は前の状態が カ ナ 状態であっても自動的に 英 数 状態となります。ＢＡＳＩＣのプログラム作成時にはプログラムの１行が終る度にこのキーをおし復帰改行を行いますから一番使用ひん度の高いキーといえます。

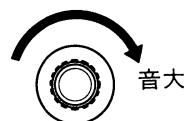
以上でキーボードの操作法を全て説明いたしましたので，次に簡単な例をとり上げ実際に文字や記号を入力してみましょう。すでにキー入力して画面がおかしくなった人は，リセットして，画面を初期状態(3ページの画面)にしてください。

リセットは，

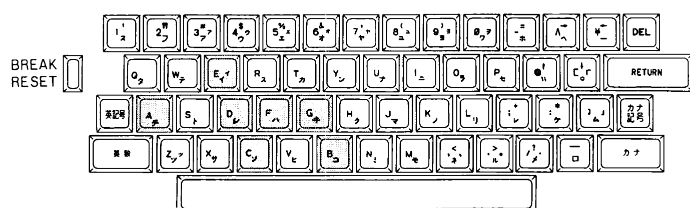
カナ記号 をおしながら BREAK RESET をおしてください。

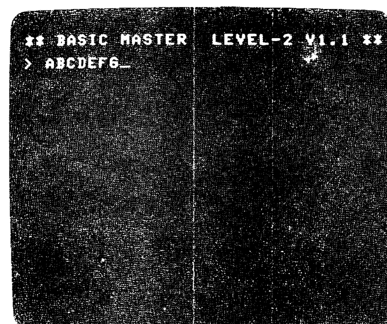
2.2 キーボードの操作方法　その2

まず，ＡＢＣＤＥＦＧと入力してみることにします。
このとき後面にある音量調節ツマミを大体中央位の位置に設定して下さい。
それでは始めます。



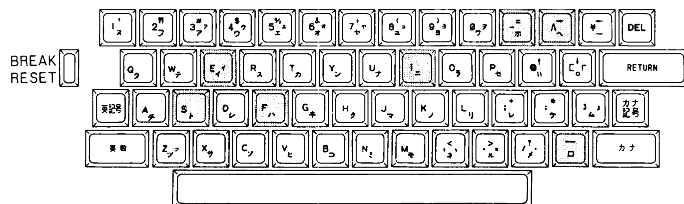
キーボードのキーでＡ～Ｇ迄の文字の入っているキーをさがし，Ａから順におしてみてください。コツコツという音（この音はキー入力が正しく行なわれていることの確認音でクリック音と呼びます。音の大きさは先ほどの音量ツマミにより変えることができます。）と共に順にカーソルが１つずつ移動しながら

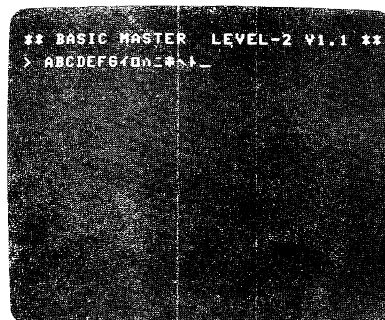




という英文字が並んだはずです。（途中で間違えたら［DEL］キーをおして訂正して下さい。）2.1 (4) で説明したように今は電源スイッチを入れた最初の状態ですから［英数］キーをおした状態が自動的に設定されており，上のように英文字が並んだわけです。

では上の文に続けてカナでイロハニホヘトと入力してみることにしましょう。カナを入力するわけですからシフトキーの［カナ］キーを１度おして下さい。（１度おしたら手をはなしてかまいません。シフトキーをおした時はカーソルは移動しません。また音もこのときは出ません。…これは［BREAK RESET］キーのときも同じです。…）そしてイ～トのキーをさがし，順におしてみて下さい。クリック音が［英数］キー状態とちがい［カナ］キー状態ではピッピッという音になります。





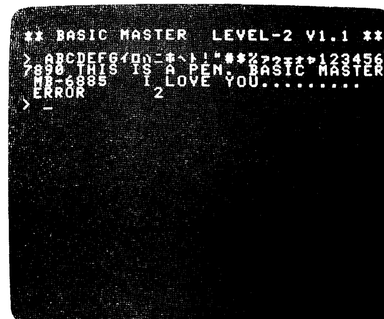
と表示されたはずです。次に続けて記号類を入力してみましょう。［英記号］キーをおしながらキーボードの左上部の！　″　＃　＄　％をおし，次に［英記号］キーから手を離し今度は［カナ記号］キーをおしながら，やはりキーボードの左上部の小文字のア　ウ　エ　オ　ヤをおしてみて下さい。

カーソル点めつ

＞ＡＢＣＤＥＦＧイロハニホヘト！″＃＄％アウエオヤ_

と並んだはずです。後は続けていろいろとご自分で好きな文字や記号を練習を兼ねて入力して見て下さい。なお0(ゼロ)とO(オー)の2つを区別するため数字の0(ゼロ)の方に斜線を入れ，Øとしています。

どんどん入力していき，三行いっぱいぐらいのところで RETURN キーをおしてみます。すると，ピーッと言う音とともに　ERROR　2　が表示されたはずです。これは,文字数が入力できる限界の79字を超えてキー入力したということを表わしています。



それでは次に79字以内で同じようにＡＢＣＤ……と入力してみます。そして同じように RETURN キーをおします。すると今度もピーッという音がして，右図のように
ＳＹＮＴＡＸ　ＥＲＲＯＲ

```
** BASIC MASTER  LEVEL-2 V1.1 **
> ABCDEFGｲﾛﾊﾆﾎﾍﾄ!"#$%ｱｳｴｵﾔ123456
7890 THIS IS A PEN. BASIC MASTER
 MB-6885   I LOVE YOU.........
 ERROR      2
> ABCDEFGHIJKLMNOPQRSTUVWXYZ
 SYNTAX ERROR
> _
```

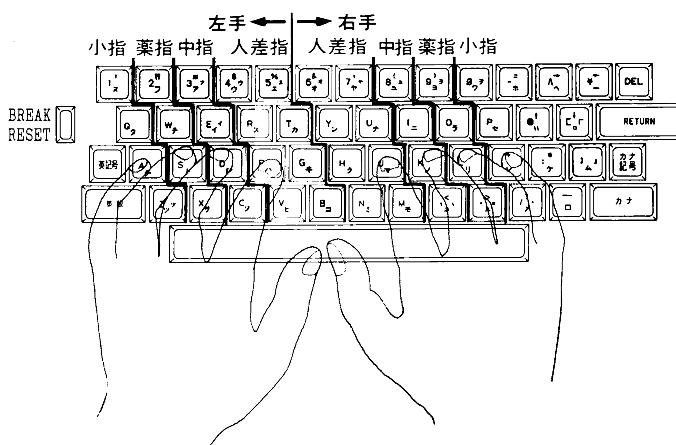
となったはずです。これは今入力した＞から始まり RETURN キーをおした所までの文章がＢＡＳＩＣの文法に沿っていないということをエラー表示して，次の行へ復帰改行したことを意味しているわけです。(これは今はＢＡＳＩＣの文法を全く気にせず単に文字や記号の入力のしかたを練習しているだけなので全くあたりまえのことなのです。…ただしもしあなたが入力した文字がＢＡＳＩＣの文法に沿った形の文章だった場合はエラーは出ず，復帰改行のみが行なわれます。…)

ともあれＳＹＮＴＡＸ　ＥＲＲＯＲなどは気にせず入力の練習を続けて下さい。画面が文字や記号で一杯になると，１行分ずつ画面の内容が上へ移動していくことも確認してください。(このように自動的に画面が上に移動することを自動スクローリングと呼びます。)

ヒント

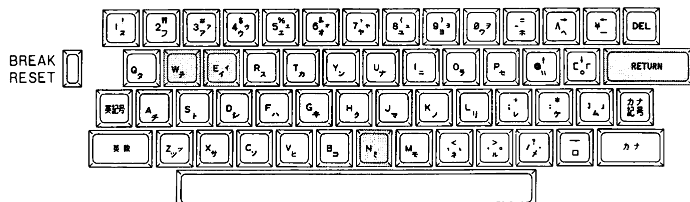
せっかくタイプライター式のキーボードがあるのですから，キーのおし方について。

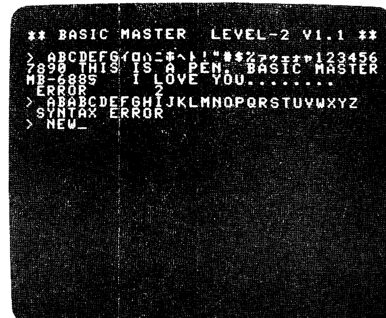
まず，左手小指を英数キーのＡに，右手小指はセミコロンにおきます。これをホームポジションと言い，この位置からどのキーがどの位置にあるかを指に覚えさせます。キーと指の関係も決っていて，図のようになっています。もちろん，こんなことにこだわらず，右手人指し指で雨だれ式に打っても一向に差し支えありません。



練習が一段落した所で，最後に1つだけベーシックマスタージュニアを本格的に働かせることをやってみましょう。大分画面が文字や記号でごちゃごちゃしていますので一度全部消してきれいにします。＞の次から下のように入力してみて下さい。(＞の次から始めるには,その前に RETURN キーをおせば良いことに練習の途中で気がつかれたかと思います。)

＞ N E W RETURN
とキー入力してください。





今まででていた文字や記号が消えて，画面の1番左上すみに＞_(カーソル点めつ)がでてきます。
音量調節ツマミが大体中央位に設定されていることを再確認してから続けて

＞ M U S I C スペース カナ ト レ ミ フ ソ ラ シ 英数 U カナ ト

とキー入力してください。正しく打ち込まれたことを確認したら(もしまちがっていたら DEL キーを使って訂正するか，ＮＥＷをキー入力する状態からやり直すかして下さい。)続けて

RETURN キーをおして下さい。ドレミファソラシドという1オクターブ分の音がベーシックマスタージュニアからでてきたはずです。上の例で入力したＮＥＷだとかＭＵＳＩＣという言葉がＢＡＳＩＣの言葉だったわけです。

それでは次の章からＢＡＳＩＣの言葉の詳しい使い方を説明するとして，ここでキーボードの操作法の説明を終ることにします。

ご注意　取扱説明書のクォーテーションマークについて

以下のページで、プログラムの例題などの中で、ＰＲＩＮＴ文などに〝　〟〈クォーテーションマーク〉が出てきます。例えば11ページの **例3** の30ＰＲＩＮＴ 〝Ａ＝〟;Ａ RETURN の部分です。〝と〟との向きのちがうクォーテーションマークが使われておりますが、キーボードからは両者共同じに扱って、4ページの②キーと51の 英記号 キーを同時におして入力します。

ヒント　キーボードに強くなろう

タイプライター式のキーボードも慣れるとこんな便利なものはないのですが，慣れないうちは，どうしてＡＢＣやアイウエオの順番に配列していないのかとつい頭にきてしまいます。しかし，これらの配列は使用ひん度や実際の使い易さを考慮して規格化されたものです。33ページ **例37** にＢＡＳＩＣを使ったキーボードの練習プログラムを掲載しますので，是非キー入力し，毎日少しずつでも練習してください。

3.BASICの使い方(V1.1)

3．1 概　要

BASICとは，人とコンピューターとが対話するときの言葉です。

ちょっと英語風，だけど単語帳はいりません。BASICの単語はわずか70語＋α，（αとはコンピューターからのメッセージが別にあるから）だからです。

かんたんな会話ならほんの3つか4つの単語でできます。

ですから，特に暗記などしなくても自然に覚えられます。それに外国語を外人に話しかけられたときのようにドキドキしなくてもいいのですから。

実は少し算数的な英語。

This is a penをBASICで表現すると

LET　T＄＝"PEN"

となります。何だか英語と算数の式とを混ぜ合せたへんてこなことばですね。ここで最初にでてくる「LET」という単語がBASICの単語，つまりキーワードです。日本語に訳すと「T＄はPENである」のあるにあたります。ベーシックマスタージュニアはこのキーワードの辞書を持っていて，あなたが話しかけてくることばをページをめくりながら探してゆき，もし辞書にでていない単語だと「余の辞書には…ということばはない！」と怒りだしてしまいます。

BASICは記号にも意味がある。

BASICは単語の数が少ないため，〈＝〉イコール，〈．〉ピリオド，〈，〉コンマなどの記号が特に重要な意味をもっています。ふだんは見落しがちな記号にも注意をはらうことが必要です。例のテンで（．で）話にならないというやつです。

BASICにも"文法"があります。

英語の文法の時間を思い出して，いやな感じを持つ人もいるかもしれませんが，これはそんなに難しいことではありません。何しろ単語が「現在形」，「過去形」などと活用しないだけでも何とありがたいことか。BASICの文法とは，たとえばこんなことです。

LET　A＝10………(日本語：Aは10だぞよ)

というのを

LET　10＝A………(日本語：？？？？………)

と言い表すことはできないといったことです。算数というか，数学というか，に強い人ならこれは納得いかないかもしれませんが，これは文法，つまり約束ごとですからしようがありません。とは言え，この程度のことですから，文法といっても特にアレルギーを起すようなことではありません。それでも，コンピューターは何しろガンコですから，こちらの方で折れないと、とても………。

ついでに先程のLET　A＝10と言ったときのAについて説明します。Aのことを変数と呼びますが，これも代数で使われる方程式の使い方とはちょっと違います。

例 1

```
10 LET  A=10 RETURN ………このときのAは10である。
20 LET  A=A+1 RETURN ……右辺のAが10, 従って左辺のAは11
30 PRINT A RETURN ………A (11) をプリントせよ。
40 END RETURN ………終り
```

といった使い方をします。代数だと２０行目の　A＝A＋１　は成立しませんが，ＢＡＳＩＣでは立派に成り立つことです。むしろこういったＢＡＳＩＣの文法をうまく使うことによってＢＡＳＩＣのプログラムがいきてきます。とにかく信じられない人は，このプログラムをキー入力して [R][U][N][RETURN] として確めてみてください。つまり，左辺のＡと右辺のＡは違うからこんな方程式（？）も成立するわけです。ですから，こんな不都合(？)と思われるＢＡＳＩＣ文ができても，ミスプリントではありませんから，念のため。

コンピューターに答えさせるのは人。

どんな立派なプログラムを作っても，それをコンピューターが表現してくれなければ何が何だか判りません。確かにコンピューターはプログラムを実行し，間違いのない答えを知っていても，その答えはＩＣという得体の知れない物の中にあるだけで，人はそれを直接みることはできません。それを表現させるのも人の役割です。つまり，こんな答えが出たら，こんな形で表現して教えてください，というプログラムをコンピューターにあらかじめ伝えておかなければなりません。ＢＡＳＩＣには，コンピューターにどんな形の返事をもらうかをたのむＰＲＩＮＴ文があり，色々な形の返事がもらえるようになっています。ＢＡＳＩＣが会話形のプログラム形式としても，コンピューターにどんなしゃべり方をさせるかは，実は使用する人の側で決めることなのです。

たとえば先程の **例１** の30行めを省略して，

例２

```
10 LET A=10[RETURN]
20 LET A=A+1[RETURN]
30 END[RETURN]
```

を実行（[R][U][N][REEURN]）しても，画面には何も現われず何をやったのか分りません。もちろん，コンピューターは答えを知っています。**例１** はもっとスマートに，

例３

```
10 LET A=10 [RETURN]
20 LET A=A+1[RETURN]
30 PRINT "A=";A[RETURN]   A＝と表示してそのあとに答え
                           をプリントせよ。
40 END [REURN]
```

として実行すると今度はA＝11と答えてくれます。

以上のように，コンピューターと言っても何から何までやってくれるわけではありません。やはり主人公は人です。人の作ったプログラムのとおりにしかコンピューターは動きません。

ＢＡＳＩＣの単語とは。

例１ **例２** **例３** で示したのがＢＡＳＩＣの文章の形（プログラム）です。プログラムの先頭には行番号がありました。そのあとにくるのがＢＡＳＩＣのステートメント，ＬＥＴ，ＰＲＩＮＴ，ＥＮＤなどです。そしてプログラムを実行させるときに，行番号なしで入力したＲＵＮなどのことを，コマンドと言っています。ベーシックマスタージュニアで使われるこれらの単語（キーワード）の一覧を次に示します。

●ベーシックマスタージュニアの機能とキーワード一覧

ベーシックマスタージュニアの機能の概要とキーワードの一覧を示します。これらの機能やキーワードの詳しい内容については，それぞれの項目で説明します。また，それぞれの項ではかんたんなプログラム例を示しますので，実際にキー入力して確実に理解してください。この一覧は実際にプログラムを作る過程で不明な点があったときの手引きとしてご利用いただき，更に詳しい内容が知りたいとき，それぞれの項のページを参照してください。各項のプログラム例では，本書の構成上，まだ説明していないキーワードを使った例もでてきますが，はじめは頁を追って一通り読んでいただき，それから改めて不明だった個所をご覧いただければ，自然にご理解いただけることと思います。

1）機能概要

項目			記号または内容	備考または例	頁
算術演算		べき乗	^	LET A=5^2	34
算術演算		加減乗除	+，−，*，／	LET B=A+(3*5/N−4)^2	34
文字および数値の比較演算			>，<，=，<> >=，<=，	IF A>=25.5 THEN 300 IF (A>15.3)*(A<=21.5) THEN 200 (15.3<A<=21.5…不可)	34 51
取り扱える数の範囲			約$\pm 3\times 10^{-39}$ 〜 約$\pm 1.7\times 10^{38}$	$\pm 2.93873588\times 10^{-39}$ 〜 $\pm 1.06338239\times 10^{38}$ 入力できる範囲 $\pm 2.93873588\times 10^{-39}$ 〜 $\pm 1.70141183\times 10^{38}$ 計算結果を表示できる範囲	17
有効桁数			最も高い精度のとき 9桁		18
行番号			1〜32767		21
マルチステートメント			：で可能	LET A=5：LET B=3	22
ダイレクト実行			可能		22
変数	数値変数	単純数値変数	A〜Z A0〜Z9		23
変数	数値変数	添字付数値変数（配列）	A(I)〜Z(I) A0(I)〜Z9(I) A(I，J)〜Z(I，J) A0(I，J)〜Z9(I，J)	1次元 2次元	25
変数	文字変数	単純文字変数	A$〜Z$ A0$〜Z9$		28
変数	文字変数	添字付文字変数（配列）	A$(I)〜Z$(I) A0$(I)〜Z9$(I)	1次元のみ	32
画面エディター			LISTされた画面上で1文字単位の編集が可能	英記号をおしたまゝで →をおしカーソルを右へ移動 ←　〃　左　〃 ↑　〃　上　〃 ↓　〃　下　〃 DELで1文字削除，他キー入力で挿入。	36

2）キーワード一覧

	キーワード	省略形	概略内容及び使用例等	頁
コマンド	NEW		前に入っていたプログラムの消去(御破算)	39
	RUN	R	プログラムの先頭から実行	39
	RUN n	Rn	行番号nから実行	39
	LIST	L	最初から最後までのプログラムリストを一気に出力する BREAK で出力停止	39
	LIST m,n	Lm,n	行番号mからnまでのプログラムリストを出力する	39
	LIST#	L#	プログラムリストをプリンタに出力する。	39
	CONTINUE	CONT または C	STOPおよび BREAK で中断されたプログラムの次の文から続けて実行する	42
	SIZE	S	プログラムの大きさ，変数領域，メモリの残りバイト数の表示	41
	LOAD"ファイル名"		プログラムの入ったテープからプログラムをベーシックマスターに移す	42,99
	SAVE"ファイル名"		ベーシックマスターに入っているプログラムをテープに移す	42,97
	VERIFY "ファイル名"		テープに入っているプログラムとベーシックマスターに入っているプログラムの比較確認	42,98
	MERGE "ファイル名"		メモリーに格納されているプログラムに，テープから読み込んだ新しいプログラムを結合する	41,99
	SEQ m,n		行番号を自動的に出力する	40
	RESEQ m,n		行番号を自動的につけかえる	40
	DEL m,n		行番号mからn番までを消去する(部分消去)	41
	MONITOR	MON	モニターにジャンプする	42
標準ステートメント	PRINT	PR または ?	画面へこのキーワードに続く内容を表示する	44
	PRINT #	PR # または ?#	プリンタへこのキーワードに続く内容を出力する	86
	INPUT	IN	キー入力の要求(RUN後キー入力を要求してくるので数値，文字を入れる。)	67
	INPUT "ストリング"	IN "ストリング"	文字を画面に表示しキー入力を要求(　同上　)	67
	LET	LET自体全てを省略可	変数に数値または文字を入れる	43
	GOTO	GO	ジャンプ	57
	IF…〔THEN〕		条件の設定〔例 IF A>1 THEN100→A>1なら行番号100へ〕	51
	FOR…TO…STEP NEXT		FORとNEXTの間をくり返すループ	61
	STOP		プログラムの実行を一時停止	71
	END		プログラムの終了	71
	GOSUB	GOS	サブルーチンへジャンプする	58
	RETURN	RET	サブルーチンの最後に入れ，サブルーチンを呼んだ文の次の文へ戻る	58
	ON…GOTO	ON…GO	ONの条件によりとび先の異なるジャンプを行なう	59
	ON…GOSUB	ON…GOS	ONの条件によりとび先の異なるサブルーチンへジャンプ	59
	READ		DATAで指定された数値および文字を変数に入れる	65
	DATA		数値，および文字をデータとして設定する	65
	RESTORE		最初のDATA文から変数に代入する	65
	DIM		配列を定義する	49
	CLEAR	CLR	表示している画面をクリアし，カーソルを画面左上に移す	68
	RANDOMIZE	RNDM	実行のたびごとに乱数の発生を不規則にする	70
	REM		プログラムをわかりやすくするためのコメントを入れる	50
	DEF FNx		ユーザー関数を定義する	69

キーワード一覧つづき

キーワード		省略形	概略内容及び使用例等	頁
特殊ステートメント	MUSIC	MU	音楽機能の指定	77
	PLOT		簡易グラフィック機能の指定	72
	POKE		絶対番地に値をいれる	74
	CALL		絶対番地の機械語サブルーチンへジャンプする	76
組み込み変数	CURSOR	CUR または !	カーソルの制御 〔例 CUR＝10, 5 カーソルを画面10,5の位置へ移動する A＝CUR 変数Aにカーソルの座標を入れる〕	90
	TIME		タイマーの制御 〔例 TIME＝0 タイマーに 0秒を設定する A＝TIME 変数Aに時間を入れる〕	91
組み込み定数	PAI		円周率 π＝3.14159265を表わす定数 〔例 A＝2＊PAIは A＝6.28318531と同じ〕	91
データの入出力ステートメント	OPEN		外部接続機器のロジカル番号を設定する	83
	CLOSE		〃 のロジカル番号を解除する	89
	PRINT#n,X	PR#n,X	ロジカル番号ｎの機器へＸを出力する	86
	INPUT#n,X	IN#n,X	ロジカル番号ｎの機器からＸへ数値又は文字を代入する	88
算術関数	SIN(X)		ＸラジアンのSinの値	93
	COS(X)		Ｘ 〃 Cos 〃	93
	TAN(X)		Ｘ 〃 Tan 〃	93
	ATN(X)		Ｘのtan^{-1}の値でラジアンで出力される	93
	EXP(X)		Ｘの指数 e^x の値	93
	LOG(X)		Ｘの自然対数$log_e X$，Ｘ＞0であること	93
	SQR(X)		Ｘの平方根の値，Ｘ≧0であること	93
	ABS(X)		Ｘの絶対値〔｜Ｘ｜〕	93
	INT(X)		Ｘをこえない最大の整数〔例 A＝INT(25.99)→25, A＝INT(−25.99)→−26〕	93
	RND(X)		疑似的な乱数の発生	93
	SGN(X)		Ｘ＞0, Ｘ＝0, Ｘ＜0のときSGN(X)＝1, 0, −1となる	93
特殊関数	PEEK(X)		Ｘで示されたメモリの絶対番地の内容をえる	75
プリント文用関数	TAB(X)		PRINT文用の関数でPRINT文がきたときのカーソル位置からＸの値だけカーソルを移動する	46
	HEX(X)		ＰＲＩＮＴ文用の関数でＸの値を16進化する	46
文字取扱い関数	LEFT$(X$, n)		文字変数Ｘ＄で表わされる文字列の左端からｎ個までの文字列を表わす	96
	RIGHT$(X$, n)		文字変数Ｘ＄で表わされる文字列の右端からｎ個までの文字列を表わす	96
	MID$(X$,m,n)		文字変数Ｘ＄で表わされる文字列のｍ個目からｎ個分の文字列を表わす	96
	LEN(X$)		文字変数Ｘ＄で表わされる文字列の長さ	96
	ASC(X$)		文字変数Ｘ＄で表わされる文字列の先頭文字のＪＩＳコードを表わす	96
	STR$(X)		数値Ｘを文字型にし，文字列として取りあつかえる様にする	96
	VAL(X$)		文字変数Ｘ＄で表わされる数を数値型にし，数値として取りあつかえる様にする	96
	CHR$(X)		Ｘで指定された文字類を表わす	96
	CURSOR$	CUR$ または !$	カーソルの位置にあらわされている文字を表わす	96
	INKEY$		キー入力された文字を表わす	96
	SPC$(X)		Ｘ個のスペースから成り立つ文字列を表わす	96

3）プリント出力の制御　（44 ページ）

指　　定	使　用　例	ＰＲＩＮＴ出力の内容
,（カンマ）	PRINT A,B	8桁毎に区切り，±の記号を含めて左づめで表示。
;（セミコロン）	PRINT A;B	±の記号を含め全て左づめで表示。
%	PRINT %n;A	Aをn桁で表示する。　11≧n≧3

4）ＬＩＳＴ出力の制御　（39 ページ）

指　　定	ＬＩＳＴ出力内容
ＬＩＳＴ	全リストを順送りで出力する。
ＬＩＳＴ n	行番号nを1行のみ出力する。
ＬＩＳＴ m，n	行番号mからnまでのリストを出力する。
ＬＩＳＴ n，	行番号nから最後までのリストを出力する。
ＬＩＳＴ ，n	行番号の最初から行番号nまでのリストを出力する。

ＢＲＥＡＫキーでリスト出力を停止

5）ＤＥＬの指定（リストの部分消去）　（41 ページ）

指　　定	ＤＥＬの内容
ＤＥＬ	何もせずSYNTAX ERRORを表示する。
ＤＥＬ n	行番号nを1行のみ消去する。（nが存在しない場合はSYNTAX ERROR）
ＤＥＬ m，n	行番号mからnまでのリストを消去する。
ＤＥＬ n，	行番号nから最後までのリストを消去する。
ＤＥＬ ，n	行番号の最初から行番号nまでのリストを消去する。

6）ＳＥＱの指定（行番号の自動出力）（40 ページ）

指　　定	ＳＥＱの内容
ＳＥＱ	行番号10から10きざみで行番号を自動的に出力する。
ＳＥＱ n	行番号nから10きざみで行番号を自動的に出力する。
ＳＥＱ n，	同　　上
ＳＥＱ n，k	行番号nからkきざみで行番号を自動的に出力する。
ＳＥＱ ，k	行番号10からkきざみで行番号を自動的に出力する。

7）ＲＥＳＥＱの指定（行番号のつけかえ）　（40 ページ）

指　　定	ＲＥＳＥＱの内容
ＲＥＳＥＱ	リストの先頭にある行番号を10に定め10きざみで行番号をつけかえる。
ＲＥＳＥＱ n	リストの先頭にある行番号をnに定め10きざみで行番号をつけかえる。
ＲＥＳＥＱ n，	同　　上
ＲＥＳＥＱ n，k	リストの先頭にある行番号をnに定めkきざみで行番号をつけかえる。
ＲＥＳＥＱ ，k	リストの先頭にある行番号を10に定めkきざみで行番号をつけかえる。

●扱える文字

プログラムを作るにあたって使用できる文字は以下の通りです。

1．英大文字	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	P	Q	R	S	T	U	V	W	X	Y	Z				
2．数字	0	1	2	3	4	5	6	7	8	9					
3．カナ文字	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ
	マ	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ヲ
	ン	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ					
4．英記号	!	″	#	$	%	&	’	(	)	=	−	^	¥	@	[
	]	+	*	,	/	<	>	?	:	;	_				
5．カナ記号	°	「	」	゛	—	、	・	。							

この分け方は，キーボードの［英 数］，［英記号］，［カ ナ］，［カナ記号］の分け方とは必ずしも同じではありません。

この他に47ページに示すようなギリシャ文字，英小文字，図形，簡単な漢字も，直接プログラムの中に書くことはできませんが，特別な命令を使って画面に出すことができます。詳しくは44ページ，PRINTステートメントの項をご参照ください。

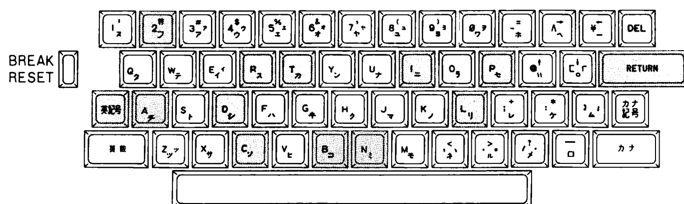
例 4　［C］［L］［R］　［RETURN］ とキー入力したのち（画面をクリアーするときのステートメントです。以下，画面がいっぱいになったら適時このステートメントを実行してください。）次のようにキー入力してみてください。

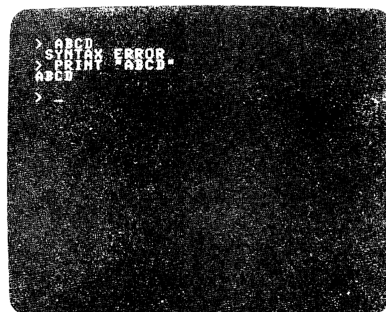
［A］［B］［C］［D］［RETURN］

これでは SYNTAX ERROR と表示されてしまいます。つぎに，

［P］［R］［I］［N］［T］［英記号］［″］（英記号を押しながら）［A］［B］［C］［D］［英記号］［″］（英記号を押しながら）［RETURN］ とキー入力してください。

今度はABCDと表示されたはずです。PRINTのあとの2つの〝　″（クォーテーションマーク）にはさまれた文字が出力されるわけです。〝　″の中には上の表にかかげた文字が全て（ただし″を除く）書きこめます。自由に試してみてください。





●扱える数の範囲

ベーシックマスタージュニアで取り扱える数の範囲は，以下の通りです。

数表示の分類		表示方法	範囲	表示の条件等	有効桁
10進数	整数表示	0～9の数字と＋－ (＋は省略できる) 例　1234	0 ≀ ±999999999	最大値をこえると，自動的に指数表示となる。 (場合によって10桁表示することもあるが有効な値は9桁まで。)	※ 9桁
	整数の後に小数がつく表示	0～9の数字と＋－． (＋は省略できる) 例　123.45	±0.00000001 ≀ ±99999999.9	最大，最小値をこえると自動的に指数表示となる。 例 0.0000000001 ↓ 1E－10	
	※ 指数表示	0～9の数字と＋－． E (＋は省略できる) 例　1.23E－20 (1.23×10^{-20}を意味する)	±2.93873588×10^{-39} ≀ ±1.70141183×10^{38}	但しキーボードから入力できる範囲は ±2.93873588×10^{-39} ≀ ±1.06338239×10^{38}	
16進数		0～9の数字とA～F 前に$をつける 例　$3F2D	$0000 ≀ $FFFF (16進数として入出力できる範囲)	PR　$FFFF+1 65536 PR HEX($FFFF+1) OVERFLOW ERROR	16進 4桁

※　次に述べる「計算の範囲と有効桁」の項を参照

例 5　Eは指数表示の記号，意味をよく理解してください。

```
10 PRINT 2.5E10
20 PRINT 2.5*10^10
30 PRINT 2.5*10000000000 ……これでは庶民にはピッタリきません。
40 END
```

例 6　マイナスの指数の意味は，

```
10 PRINT 1.33E-10
20 PRINT 1.33*10^(-10) ………………－（マイナス）は（　）でくくる。
30 PRINT 1.33/10000000000
40 END
```

●計算の範囲と有効桁

(1)計算結果の表示範囲と使用できる範囲

1．PRINT文やINPUT文で使用できる数の範囲は

$\pm 2.93873588 \times 10^{-39}$
〜　　※
$\pm 1.06338239 \times 10^{38}$

2．計算結果を表示できる数の範囲

$\pm 2.93873588 \times 10^{-39}$
〜　　※
$\pm 1.70141183 \times 10^{38}$

※　使用できる範囲と表示できる範囲の違い

例　PRINT　1．06338240E38
　OVERFLOW　ERROR　と表示される

例　PRINT　1．06338239E38＋．6380294 4E38
　1．70141183E38　（E38 ← 使用できる範囲；1.70141183E38 ← 表示できる範囲）

3．関数の計算できる範囲は，93ページに詳しく示してあります。

(2)OVERFLOW と UNDERFLOW

	処　理	例
OVERFLOW （計算途中，または結果が±1.70141183×10^{38} をこえた場合。）	発生時点で OVERFLOW ERROR を表示し，そのOVERFLOWした値に最大値 $\pm 1.70141184 \times 10^{38}$ を設定しプログラムを続行する。 但し乗算結果のOVERFLOWは発生時点でERROR表示をし実行を中断する。	10　A＝1．063E38 20　B＝1．062E38 30　C＝（A＋B）／2 40　PR　A＋B，C 実行すると OVERFLOW ERROR←行番号30のERROR OVERFLOW ERROR←行番号40のERROR 1.70141184E38　8.50705918E37 （A＋Bの結果）　（Cの結果） 30　C＝A＊B　と変更しこれを実行すると OVERFLOW ERROR 　30　LET　C＝A＊B　となる。
UNDERFLOW （計算途中，または結果が±2.93873588×10^{-39} を下まわる場合。）	発生時点でUNDERFLOWした値に0を設定しプログラムを続行する。※※ （UNDERFLOW ERRORの表示はしない。!!）	10　A＝2．93873588E－39 20　C＝A／10 30　PR　A，C 実行すると 　2．93873588E－39　　0 （Aの結果）　（Cの結果）

※※0に設定する理由は，$\pm 2.93873588 \times 10^{-39}$ よりも小さい値が極めて0に近い値であると云う考え方をとるためです。

(3)計算の範囲と有効桁

ベーシックマスタージュニアは演算結果を有効桁数９桁まで正確に表示することができますが，演算を行う数値が極端に大きなときあるいは極端に小さなとき，演算結果の下位の桁が正しく表示されないことがあります。これは，数値の記憶・演算を行うときに下位の桁を近似処理するためです。

ａ．ＰＲＩＮＴ文やＩＮＰＵＴ文などで整数10桁を使用し，整数表示で10桁表示した場合，下位桁が狂うことがあります。なるべく９桁以内あるいは11桁以上を使用するようにしてください。

・計算結果で整数表示10桁を行った場合９桁までが有効

例 7

```
PRINT 5555555555;6666666666 RETURN
```

この場合，５５５５５５５５５８　６６６６６６６６７０　と表示される。

例 8

```
PRINT 1234567890*8 RETURN
```

この場合，９８７６５４３１２４　と表示され，10桁めは無効。

ｂ．べき乗の計算は，ＬＯＧとＥＸＰの関数を使って計算するため，有効桁が小さくなります。

例 9　$(2^{\frac{1}{N}})^N=2$　ですがＮが大きくなると誤差が大きくなります。

```
A=2^(1/1200):PRINT A^1200 RETURN
```

この場合，２．０００００００６３　となります。

ｃ．小数点表示の演算結果で有効桁が８桁になることがあります。このような場合，８桁以内の表示の指定をすれば，９桁めが４捨５入されます。

例 10

```
A=2.46-2:PRINT A:PRINT %8;A RETURN
```

この場合，　．４６０００００００１
　　　　　　．４６　　　　　　　　となります。

ｄ．三角関数　ＳＩＮ（Ｘ），ＣＯＳ（Ｘ），ＴＡＮ（Ｘ）は周期関数ですが，
　$X \geqq 10^5$ 以上になると，有効桁が順次小さくなります。

例 11　三角関数は角度を（　）でくくって計算します。

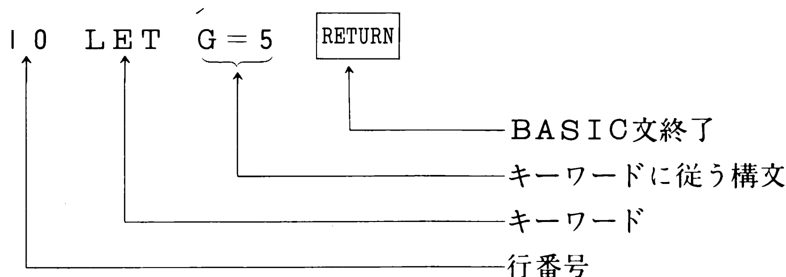
```
10 PRINT SIN(PAI) ……………… SIN(PAI)=Sin 180°=0
20 PRINT SIN(1000*PAI) ……………… ラジアン単位です。
30 PRINT SIN(10000*PAI)
40 END
```

●BASICのステートメント

何かを実行するために，コンピューターと対話する言葉がＢＡＳＩＣです。ＢＡＳＩＣによるプログラムは，ステートメントの集まりです。

1．ＢＡＳＩＣのプログラムはステートメントの集まったものである。

2．ＢＡＳＩＣ文の一行は，行番号，キーワード，キーワードに従った構文によって構成され，RETURN キー入力によって終了する。



例外１，ダイレクト実行（22ページ参照）では行番号がない。

例外２，キーワードＬＥＴは省略してもよい。

3．行番号の若い順に実行する。

4．プログラムの作成時，いかなる行番号の順に入力してもよい。

（ベーシックマスタージュニアは自動的に行番号の若い順に並べかえて実行する。）

5．ＢＡＳＩＣ文の一行には行番号も含めて79字まで入力できます。

例 12　はじめに　＞ N E W　RETURN　とキー入力したのち（新しくプログラムを入力するときのコマンドです。以下の例では省略）つぎの例をキー入力してみてください。

３０　ＥＮＤ RETURN ……………… プログラム終了。

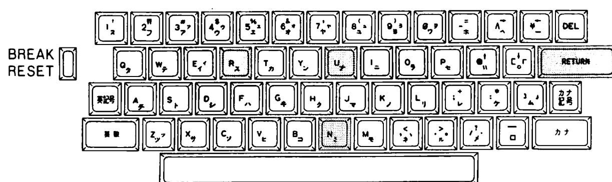
２０　ＰＲＩＮＴ　Ａ RETURN ………… Ａをプリントせよ。

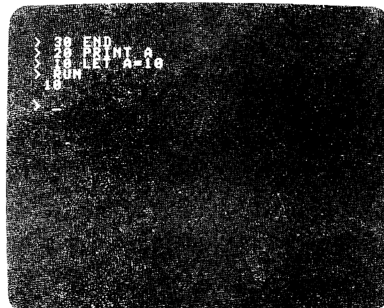
１０　ＬＥＴ　Ａ＝１０ RETURN ……… Ａに１０を代入せよ。（Ａは１０である。）

このプログラムを実行するには，

R U N　RETURN　とキー入力してください。

右図のように表示されるはずです。





●行番号について

プログラムは何行かのＢＡＳＩＣ文の集まりです。この各行の先頭に行番号を書きます。

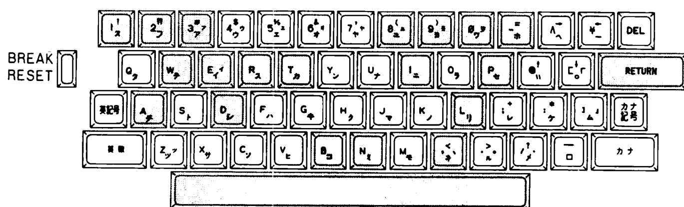
> １．行番号は１～３２７６７の正の整数。
> ２．行番号の前に入れたスペースは，ないものとみなす。
> ３．ＢＡＳＩＣは行番号の値が増える順序で各行を実行する。
> 　（但し，ＧＯＴＯ，ＧＯＳＵＢなどによるジャンプは例外。）
> ４．行番号は，とび番号であってもよい。

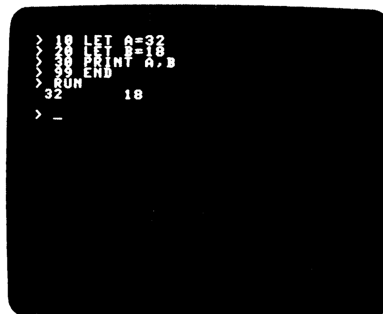
例13　[N][E][W] [RETURN] とキー入力したあと，つぎの例を実行してみてください。

```
10␣LET␣A=32 [RETURN] ……………… Aに32を代入せよ（Aは32です。）
20␣LET␣B=18 [RETURN] ……………… Bに18を代入せよ（Bは18です。）
30␣PRINT␣A，B [RETURN] …………… AとBをプリントしなさい。
99␣END [RETURN] …………………………… プログラムはこれで終り。
```

␣：スペースキーを押すことを表しています。

番号はとんでもよい。





このプログラムを実行するには，

[R][U][N] [RETURN]

とキー入力してください。画面に右図のように表示されるはずです。

30行目の，ＰＲＩＮＴ　Ａ，Ｂの，（コンマ）は表示されません。このコンマはＡとＢとが画面にプリントされるときの間隔を示す記号です。詳しくは45ページをご参照下さい。

ヒント　行番号のつけ方

１．行番号は１０きざみにつけると，後で行を挿入したいとき，大変らくになります。（間に９行挿入できます。）

２．ＥＮＤ文の行番号は，９９９９とか３００００とかの大きな数にしておくと，あとで行を追加してもＥＮＤの行は修正しなくてもすみます。

３．ＧＯＳＵＢでジャンプする先の行番号も始めから５００とか１０００とかの覚えやすい大きい番号にしておけば，後で行番号が不足して修正するなどという目にあわずにすみます。

　何しろ，行番号は３２７６７まで使えます。ケチケチせず思いきった行番号をつけてください。

●マルチステートメント

１．：（コロン）で区切ることによって，一つの行番号のもとに複数個のステートメントを書くことができる。

ただし，以下のステートメントにつづくマルチステートメントは許されない。

〈行番号〉　ＧＯＴＯ〈行番号〉：-----------------
〈行番号〉　ＧＯＳＵＢ〈行番号〉：-----------------
〈行番号〉　ＯＮ--------ＧＯＴＯ〈行番号〉：-----------------
〈行番号〉　ＯＮ--------ＧＯＳＵＢ〈行番号〉：-----------------
〈行番号〉　ＥＮＤ：-----------------
〈行番号〉　ＤＥＦ　ＦＮＡ--------：-----------------
　　　　　この場合は：以下は無視される。但しＲＥＭ文はかける
〈行番号〉　ＤＡＴＡ　○，○，○，：-----------------
　　　　　この場合，：------はＤＡＴＡとみなされる。

２．マルチステートメントで書かれた複数個のステートメントは〈行番号〉に近い方から順に実行する。

例14　マルチステートメントの例，はじめに［N］［E］［W］［RETURN］

```
１０　ＬＥＴ　Ａ＝３：ＬＥＴ　Ｂ＝５：ＰＲＩＮＴ　Ａ，Ｂ　[RETURN]
１００　ＥＮＤ　[RETURN]
```

［R］［U］［N］［RETURN］で確めてください。

●ダイレクト実行

１．行番号なしでキー入力したステートメントをただちに実行する。
２．ダイレクト実行されたステートメントはプログラムとして登録されない。

例15

```
ＰＲＩＮＴ　〝ダイレクト〞，１０／３＊３　[RETURN]
```

例16

```
Ａ＝２．５：Ｂ＝５：ＰＲＩＮＴ　Ａ＊Ｂ　[RETURN]
```
マルチステートメントも可能

例17

```
１０　ＰＲＩＮＴ　Ａ，ＳＱＲ（Ａ），Ａ＾２
１００　ＥＮＤ
```

```
Ａ＝２．５：ＧＯＴＯ　１０　[RETURN]
```
------たとえば上記プログラムの10行目にダイレクト実行でとび込むことが可能，デバックのときなどに有効な方法です。

3.2 構文の要素／●変数について

プログラムの中で，適当な数値あるいは文字列（１字以上の文字の並び）などを代入できる文字を変数と呼びます。

１．分類

- 変数
  - 数値変数
    - 単純数値変数　例）　Ａ＝Ｂ＊３
    - 添字付数値変数（配列）
      - １次元　例）　Ａ（Ｉ）＝Ｃ＊Ｉ
      - ２次元　例）　Ａ（Ｉ，Ｊ）＝２５＊Ｉ
  - 文字変数
    - 単純文字変数　例）　Ａ＄＝〝イロハ〃
    - 添字付文字変数（１次元のみ）（配列）　例）　Ａ＄（Ｋ）＝〝ＡＢＣ〃

２．変数として使用できる文字はＡ ～ Ｚまでの英大文字１字の２６種と，英大文字１字とそれに続く０～９までの数字からなるＡ０～Ｚ９の２６０種，計２８６種。

３．文字変数は後に＄をつけて数値変数と区別する。

例）　Ａ５＝３．１４　　Ａ５は数値変数

　　　Ａ５＄＝〝ヘンスウ〃　　Ａ５＄は文字変数

全ての変数は，プログラムの実行される順番でメモリー（ＲＡＭ）の最終番地から，格納領域が確保されます。詳しくはＳＩＺＥコマンドの項をご参照ください。

●数値変数

プログラムの中で，適当な数値などを代入することのできる文字を数値変数と呼びます。

１．数値変数には単純数値変数と添字付数値変数とがある。

　　（添字付数値変数：次節で詳しく説明）

２．数値変数として使用できる文字はＡ ～ ＺとＡ０～Ｚ９の計２８６種。

３．数値変数には，定数，式および関数の値を代入できる。

〈数値変数〉＝定数　　例）Ａ＝５．４６（定数とは数値で表わされる値）

〈数値変数〉＝数値変数　　例）Ａ＝Ｂ

〈数値変数〉＝式　　例）Ａ＝３＋１５，Ｙ＝３＊Ｘ＋５＾２

〈数値変数〉＝関数　　例）Ａ＝ＴＡＮ（Ｘ），Ｂ＝ＳＧＮ（Ｘ）

例 18　２次方程式は次のように記述できます。

```
１０　ＬＥＴ　Ａ＝２ ----------------------- ┐
２０　ＬＥＴ　Ｂ＝５                         │
３０　ＬＥＴ　Ｃ＝１７                       ├変数A, B, C, Xに定数を代入
４０　ＬＥＴ　Ｘ＝０ ----------------------- ┘
５０　ＬＥＴ　Ｙ＝Ａ＊Ｘ＊Ｘ＋Ｂ＊Ｘ＋Ｃ ----------Ｙ＝ＡＸ²　＋ＢＸ＋Ｃ
６０　ＰＲＩＮＴ　Ｙ
７０　ＬＥＴ　Ｘ＝Ｘ＋１ ------------------------前のＸの値に１を加える。
８０　ＩＦ　Ｘ＜１０　ＧＯＴＯ　５０ ----------７０行で１を加えて１０より小さ
                                                  かったら５０行からもう一度実行。
９０　ＥＮＤ ------------------Ｘが１０だったら終了。
```

例 19　半径から円周と面積を求めるプログラム例です。

```
１０　ＬＥＴ　Ｒ＝１５ -----------------------Ｒ（半径）は１５です。
２０　ＬＥＴ　Ｌ＝２＊ＰＡＩ＊Ｒ --------------Ｌ（円周）は２×πＲです。
３０　ＬＥＴ　Ｓ＝ＰＡＩ＊Ｒ＊Ｒ --------------Ｓ（面積）はπ×R²です。
４０　ＰＲＩＮＴ　Ｒ；Ｌ；Ｓ ------------------Ｒ（半径），Ｌ（円周），Ｓ（面積）をプリントせよ。
１００　ＥＮＤ ---------------------------------プログラムは終り。
```

[R][U][N] [RETURN] で実行してみてください。つぎに，このプログラムをこのままにしておき（[N][E][W] [RETURN] としなければよい。）少しプログラムを変更します。

```
１０　ＩＮＰＵＴ　Ｒ [RETURN] ----------------Ｒ（半径）を入力し，20行，30行のＲに代入。
```

これで１０行めのステートメントが変更されています。(エディター機能)

[R][U][N] [RETURN] で実行してください。１行下に？が表示され，カーソルが点めつします。そこで

[1][5] [RETURN] とキー入力してください。さっきと同じ答えが画面にプリントされるはずです。もっと他の半径のときの円周を知りたいときはもう一度 [R][U][N] とすればよいのです。

ヒント　変数を何にするか

長さの計算ならＬ，重さならＷなど変数に選ぶアルファベットはプログラムで何をやっているかが後ですぐ判断できるものを選ぶのも一つの手です。

また，将来共用するようなサブルーチン※を行番号の大きい所にまとめて格納するようなとき，サブルーチンに使用する変数と，メインプログラムで使う変数とがダブってしまうと不都合がおきることがあります。そのようなことを防ぐために，サブルーチン用の変数をあらかじめ予約しておき，メインプログラムではその変数は使わないようにすることも考えられます。

※サブルーチンについては，58ページをご参照ください。

●添字付数値変数（数値の配列）

カッコでくくった添字をもつ数値変数で，添字を定数，数値変数，式などで定めることができ，変数の使用範囲を広げることができるものです。

●２３ページ数値変数の文法に従う外に以下の文法が追加されます。

１．添字付数値変数はＡ～Ｚ，Ａ０～Ｚ９までの２８６種の文字が使える。
２．１次元，２次元両方の配列が使える。
　　１次元の配列　Ａ（ｎ）………Ａ（１）からＡ（ｎ）までｎ個の変数を定義。
　　２次元の配列　Ａ（ｍ，ｎ）…Ａ（１，１）からＡ（ｍ，ｎ）までのｍ×ｎ個の変数を定義。
３．添字は〈定数〉，〈変数〉，〈式〉のいずれであってもよい。
　　Ａ（〈定数〉），Ａ（〈定数〉，〈定数〉）……Ａ（５），Ａ（５，８）
　　Ａ（〈変数〉），Ａ（〈変数〉，〈変数〉）……Ａ（Ｂ），Ａ（Ｂ，Ｃ）
　　Ａ（〈式〉），Ａ（〈式〉，〈式〉）……Ａ（３＊Ｂ），Ａ（４＋Ｂ，Ｃ／３）
４．添字付数値変数を使用するときは，プログラムの先頭でキーワードＤＩＭ※を使って，使用する最大配列を宣言しなければならない。
　　例）１０　ＤＩＭ　Ａ（５）　，　１０　ＤＩＭ　Ａ（５，８）
　　（５，８の数字：定数，変数，式の最大値）
５．一度配列変数として宣言した文字は，別の数値変数として使えない。
　　※49ページ参照

後述する添字付文字変数（文字の配列）とは重複して使えます。（例１０　Ａ＝３：Ａ＄＝〝イ〟）

例20　例題が難しく思われる場合，読みとばしてあとで試してください。

```
 10 DIM  A(5)                  ……配列宣言，最大5個の配列を使用しますヨ。
 20 LET  S=0                   ……総和を求めるときに使う手です。
 30 LET  B=0
 40 LET  B=B+1
 50 PRINT "A(";B;")=";         ……セミコロン
 60 INPUT A(B)
 70 LET  S=S+A(B)
 80 PRINT                      ……1行あける。
 90 IF   B<5 GOTO 40
100 PRINT "A(TOTAL)=";S
130 PRINT "A(AVERAGE)=";S/B    ……AVERAGE：ヘイキンなどとカナでもよい。
500 END
```

このプログラムは，添字付数値変数の添字を４０行，９０行のループの中で変数Ｂを使って１つずつ増加させている例です。Ａ（１）＝？と表示が出たら数字をキー入力して RETURN とし，Ａ（１）からＡ（５）までを入力すると合計と平均が得られます。

例21　こんどは２次元の配列を使ったプログラム例です。**例20**の変形です。

```
 10  DIM A（4，12）………………………配列宣言，X：横軸，Y：縦軸。
 20  FOR Y＝2 TO 12………………………縦軸11例
 30  FOR X＝1 TO 4………………………横軸4行
 40  A（X，1）＝X………………………初期値の設定
 50  A（X，Y）＝A（X，Y－1）＊2
 60  PRINT A（X，Y－1），………………最後のコンマを必ず入れる。
 70  NEXT X………………………………横軸のループ，30行に戻る。
 80  PRINT…………………………………ここで行がえをします。
 90  NEXT Y………………………………縦軸のループ，20行に戻る。
500  END
```

例22　同じく２次元の配列を使ったプログラム例です。Ｙ１～Ｙ６球団の試合数と勝敗をキー入力すると，負け数と勝率が表示されます。表作りのため，ＣＵＲＳＯＲステートメントを多用しています。例題が難しいと思われるときは読みとばしてあとで試してください。理解できた人は，勝率による並べかえに挑戦してください。

```
 10  DIM A（4，6）………………………配列宣言，横4行，縦6列
 20  CLEAR…………………………………画面をクリアー
 30  FOR Y＝1 TO 6………………………縦のループ
 40  LET CURSOR＝4，10
 50  PRINT "Y"；Y；"チーム ノ シアイス
     ウハ"；：INPUT A（1，Y）…………試合数入力
 60  LET CURSOR＝13，12
 70  INPUT "カチスウハ "，A（2，Y）：……………勝数入力
     CLEAR
 80  LET A（3，Y）＝A（1，Y）－A（2，Y）…………負け数
 90  LET A（4，Y）＝INT（A（2，Y）／
     A（1，Y）＊1000）／1000…………3桁の勝率計算。INT（X）は
100  PRINT CHR$（7）：NEXT Y                       Xの整数。
110  CLEAR
120  PRINT " チーム␣␣シアイスウ␣カチス
     ウ␣マケスウ␣␣ショウリツ"…………表づくりの開始
130  FOR Y＝1 TO 6
140  LET CURSOR＝1，Y＊3：PRINT
     "Y"；Y………………………………文字列の配列により実名を入れることも
                                        可能です。
150  RESTORE……………………………データの始めから変数Tに代入。
160  FOR X＝1 TO 4
170  READ T……………………………Tは横軸のカーソル位置のデータ
180  LET CURSOR＝T，Y＊3：PRINT            （200行）
     A（X，Y）；
190  NEXT X：NEXT Y
200  DATA 6，12，17，22
210  END
```

配列の簡単な考え方

1次元配列とか2次元配列とかの考え方は，次の様に考えるとわかりやすいでしょう。

1次元配列例

```
10  DIM A(5)……………Aという変数に最大5個の記憶場所を与える。
20  LET A(1)=10……………A(1)という場所に10という数値をしまう。
```

A(1)
A(2)
A(3)
A(4)
A(5)

2次元配列例

```
10  DIM B(5, 3)……………Bという変数に最大5×3=15個の記憶場所を与える。
20  LET B(3, 2)=10…………B( 3 , 2 )という場所に10という数値をしまう。
                             ⋮   ⋮
                          3行目 2列目
```

	1列目	2列目	3列目
1行目	B(1,1)	B(1,2)	B(1,3)
2行目	B(2,1)	B(2,2)	B(2,3)
3行目	B(3,1)	[redacted]	B(3,3)
4行目	B(4,1)	B(4,2)	B(4,3)
5行目	B(5,1)	B(5,2)	B(5,3)

●文字変数

プログラムの中で，適当な文字列（１字以上の文字の並び）を代入することのできる文字を文字変数と呼びます。

１．文字変数には，単純文字変数と添字付文字変数とがある。
　　　　　　　　　　　　　　↑――次節で詳しく説明

２．文字変数として使用できる文字はＡ～Ｚ，Ａ０～Ｚ９までの２８６種類で，これらの文字の後に必ず＄をつける。

３．文字変数には，〝　〟で囲んだ文字列，文字変数および＄のついた文字取り扱い関数を代入できる。

例）Ａ＄＝〝モジ　ヘンスウ〟…………Ａ＄が「モジヘンスウ」という文字列を表わす。

４．〝　〟で囲った文字列は，１つの文字変数あたり最大３２字まで代入できる。

例）Ｂ＄＝〝モジ　ヘンスウノ　カズ　ハ…………Ｚ〟
　　　　|←――――最大３２字まで――――→|

（３２字をこえて指定しても，最初から３２文字までを代入し残りは切り捨てる。）

５．文字変数，〝　〟で囲った文字列および＄のついた文字取り扱い関数を互に，＋（プラス）の記号で結合することができる。

例）Ａ＄＝〝セイベ　ツ〟
　　Ｂ＄＝〝オトコ〟
　　Ｃ＄＝〝オンナ〟
　　Ｄ＄＝Ａ＄＋〝　〟＋Ｂ＄…………Ｄ＄＝〝セイベ　ツ　オトコ〟と同じ
　　Ｅ＄＝Ａ＄＋〝　〟＋Ｃ＄…………Ｅ＄＝〝セイベ　ツ　オンナ〟と同じ

６．文字変数および＄のついた文字取り扱い関数は，関係演算子で大小比較（47ページの文字図形の表示コードに出てくる文字のコード比較）ができる。

文字列の長さが違う場合，同じ長さの部分のみの比較を行い，長さそのものの比較は行なわない。

例）Ａ＄＝〝ＡＢＣＤＥ〟：Ｂ＄＝〝ＡＢＢ〟のとき，
　　３０　ＩＦ　Ａ＄＞Ｂ＄　ＧＯＴＯ　５０　のような比較を行なうと，
　　～～～の部分のみの比較を行ない，Ａ＄のＤＥはないものとする。

［上記の例ではＡ＄とＢ＄の２字めまでは等しく，３字めのＣとＢとではＣの方が大きい（Ｂは＄４２，Ｃは＄４３）ので，３０行の条件式は成立し，５０行に分岐します。］

例）Ａ＄＝〝ＡＢＣＤＥ〟：Ｂ＄＝〝ＡＢＣ〟を比較した場合，Ａ＄と，Ｂ＄とは等しいものと判定されます。

例23 文字図形の表示コードを文字変数に代入することができます。

```
10 CLEAR ………………………… 画面をクリアーしてカーソルを左上端に。
20 LET A$=CHR$ ($41) ……… CHR$ (X) は，47ページのコード表よりさがします。
30 LET A$=A$+CHR$ ($42)
40 PRINT A$ ………………… +は加算ではなく，結合を意味する。
50 END
```

例24 文字列を文字変数に代入するときは，必ず〝 〞で囲う。

```
10 LET A0$=〝I_〞:LET A1$=〝YOU_〞:
   LET A2$=〝LOVE_〞
20 LET A3$=〝DO_〞:LET A4$=〝ME_〞:
   LET A5$=〝.〞:LET A6$=〝?〞
30 CLEAR
40 PRINT A0$;A2$;A1$;A5$…セミコロンでなく+記号でもOK。
50 PRINT
60 PRINT A3$;A1$;A2$;A4$;A6$
70 END
```

例25 ここから文字取扱い関数の説明例です。文字取扱い関数については96ページにまとめの表があります。

```
10 LET A$=〝BASIC-MASTER〞
20 CLEAR
30 LET L=LEN (A$) ……………… LEN(X)は文字の長さを表わす。
40 FOR I=1 TO L
50 PRINT I,LEFT$ (A$,I) …LEFT$(X$,n)は左から順番に
60 NEXT I                  文字を取扱うときの関数です。
70 END
```

例26 文字列を，左から順に，まん中を，右から順にと部分的に取扱うことができます。

```
10 CLEAR
20 LET A$=〝ABC〞
30 PRINT LEFT$ (A$,1)
40 PRINT MID$ (A$,2,1)
50 PRINT RIGHT$ (A$,1)
60 END
```

ヒント　文字取扱い関数について。　数値形と文字形

数値形　LEN (A$) のようにLENと ( ) との間に$がない。
文字形　LEFT$ (A$) のようにLEFTと ( ) との間に$がある。
　数値形と文字形は全く性質の違うものです。

例27

```
                   5     5     5
10 CLEAR
20 LET A$="LEFT_MID__RIGHT"
30 FOR I=0 TO 2
40 PRINT MID$(A$, I*5+1, 5)
50 NEXT I                 ↑─Iが0なら1, 1なら6, 2なら11となる。
60 END
```

例28 LEN(X$)は数値形。数字と同じように計算ができます。

```
10 CLEAR
20 INPUT "モジ レツ", A$
30 PRINT
40 IF LEN(A$)/2<>INT(LEN(A$)/2) …INT(X)は
   THEN LET A$=A$+" "   小数点を切りすてて,整数部だけを取り出す。
50 FOR I=1 TO LEN(A$)/2+1
60 PRINT MID$(A$, I, LEN(A$)/2)
70 NEXT I           ↑─I番めから ↑─この数だけがMID$の文字列
80 END
```

例29 STR$(数値)は，数値を文字形として取扱えるようにする関数

```
10 LET A=1234:LET B=5678
20 LET A$=STR$(A):PRINT A$
30 REM A$="_1234" ………………… 1234は+1234とプ
                                 ラスが省略されている。
40 LET B$=STR$(B):PRINT B$
50 REM B$="_5678" ………………… LEN(B$)=5となり
                                 ます。
60 LET C$=A$+B$:PRINT C$
70 REM C$="_1234_5678"
80 END
```

例30 従って，STR$(X)の先頭のスペースを処理するには

```
10 INPUT "A", A…………数値入力要求
20 LET A$=STR$(A)
30 LET A$=RIGHT$(A$, LEN(A$)-1)……先頭のスペー
40 PRINT A;A                         スをとる。
50 PRINT A$;A$
60 END
```

例31 ＳＴＲ＄と反対の処理をするのがＶＡＬ（Ｘ＄)。数値形。
ダイレクト実行で確めてください。

```
PRINT VAL ("1234 56")
PRINT VAL ("200YEN") *1200
```

例32 ＳＰＣ＄(Ｘ)はＸ個のスペースから成る文字列を表わします。

```
10 LET A$="12345" :LET B$="6789" …文字列の数字
20 LET A=VAL (A$) :LET B=VAL (B$)
30 LET C$=A$+SPC$(10)+B$:LET C=A+B…C$はA$とスペース10個とB$との結合からなる文字列で,(+) は結合記号です。
40 PRINT                                C$=A$*SPC$(10)*B$はできません。
50 PRINT A$, B$, C$
60 PRINT
70 PRINT A, B, C
80 END
```

例33 ＡＳＣ（Ａ＄）は先頭の文字を文字コード表の数値にする。

```
10 CLEAR
20 LET A$="AB"
30 PRINT A$, HEX (ASC (A$)) ;…HEX(X)は4桁16進出力
40 PRINT HEX (ASC (RIGHT$ (A$, 1))) …右側の文字すなわち"B"
50 END
```

例34 ＰＥＥＫ（Ｘ）関数の項，モニター編（モニターのＤコマンド）を先にご覧ください。

```
10 CLR :IN "START␣", S, "␣END␣",
   E:S1=S: S0=0:PRINT
20 D0$="0123456789ABCDEF"
30 FOR J=1 TO INT ((E-S) /8) +1
40 PR HEX (S1) ;
50 FOR I=S1 TO S1+7
60 D=PEEK (I) : S0=S0+D
70 D1=INT (D/16) : D2=D-D1*16
80 D1$=MID$ (D0$, D1+1, 1)
90 D2$=MID$ (D0$, D2+1, 1)
100 D$=D1$+D2$+"␣" :PR D$;
110 NEXT I: S1=S1+8:PR
120 IF INT (J/16) <>J/16 GO 150
130 PR :PR "CHECK SUM=" ;HEX (S0) :
    S0=0:PR
140 A$=INKEY$:IF A$<CHR$ (1) GO 140
150 NEXT J
160 PR :PR "CHECK SUM= ";HEX (S0) :END
```

●添字付文字変数（文字の配列）

カッコでくくった添字をもつ文字変数で，添字を定数，数値変数，式などで定めることができ，変数の使用範囲を広げることができるものです。

添字付文字変数は文字変数の一種ですから，２８ページの文字変数の文法に当然従うほか，以下の文法が追加されます。

１．添字付文字変数として使用できる文字はＡ ～ Ｚ，Ａ０ ～ Ｚ９の計２８６種で，これらの文字の後に必ず＄をつける。

　例）　Ａ＄（Ｉ），Ｂ０＄（Ｉ）

２．１次元の配列のみ使える。　　（添字付数値変数は２次元も使える。）

　例）　Ａ＄（ｎ）………Ａ＄（１）からＡ＄（ｎ）までのｎ個の変数を定義。

３．添字は〈定数〉，〈数値変数〉，〈式〉のいずれであってもよい。

　例）　Ａ＄（〈定数〉）…………………Ａ＄（５）
　例）　Ａ＄（〈数値変数〉）……………Ａ＄（Ｘ）
　例）　Ａ＄（〈式〉）……………………Ａ＄（Ｉ＋３＊Ｊ）

４．添字付文字変数を使用するときは，プログラムの先頭でキーワードＤＩＭを使って，使用する最大配列を宣言しなければならない。

　例）　１０　ＤＩＭ　Ａ＄（１５）…………最大１５個の配列を使用する。

５．一度添字付文字変数（文字の配列）として宣言した文字は別の文字変数として使えない。

例35　普通の文字変数では，１つの文字列に１つの文字変数が対応する。

```
10 LET A0$="オダ ノブ ナガ "
20 LET A1$="トヨトミ ヒデヨシ "
30 LET A3$="トクガワ イエヤス "
40 PRINT A0$;A1$;A3$
50 END
```

例36　添字付文字変数は，１種類の変数で処理でき，ＩＮＰＵＴやＰＲＩＮＴを，添字で指定できる。

```
10 DIM A1$(100) ………………… 配列宣言　最大100個の文字列を使用。
20 FOR I=1 TO 100 ……┐
30 INPUT A1$(I)        ├ 100個の文字列を入力する。
40 NEXT I ……………………┘
50 INPUT K:PRINT A1$(K) …… 番号（1～100）を入力し，
60 GOTO 50                    その番号に対応する文字列を出力する。
```

例37 キーボードの練習をするプログラムです。間違いがすくなく，早く打てるように。毎日少しずつ，プログラムでつかれたときにやってください。1ヶ月もすればプロ級の腕前になるとかならないとか。

```
10 DIM A$(3)
20 CLEAR
30 LET A$(1)="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
40 LET A$(2)="アイウエオカキクケコサシスセソタチツテトナニヌネノ"
50 LET A$(3)="ハヒフヘホマミムメモヤイユエヨラリルレロワイウエヲン"
60 PRINT "     キーボード ニ ツヨクナロウ !!!"
70 PRINT "     ---------------------"
80 PRINT
90 PRINT " ABC ノ アルファベット デスカ・・・     1"
100 PRINT " アイウ ・・・・・ ナニヌネノ デスカ     2"
110 PRINT " ハヒフ ・・・・・ ン デスカ     3"
120 PRINT :INPUT "      バンゴウ ヲ エランデ クダサイ",K
130 PRINT
140 PRINT "ピーット イウ オトガ ナッタラ ハジメテクダサイ"
150 FOR I=1 TO 1500:NEXT I
160 CLEAR
170 LET CURSOR=1,10:PRINT A$(K)
180 MUSIC Uシ
190 LET TIME=0
200 FOR N=1 TO LEN(A$(K))
210 LET B$=MID$(A$(K),N,1)
220 LET C$=INKEY$:IF C$<=CHR$($F) THEN GOTO 220
230 IF C$<>B$ THEN GOTO 220
240 LET CURSOR=N,10:PRINT "␣"
250 MUSIC P0シ
260 NEXT N
270 PRINT
280 LET T=TIME
290 MUSIC P0トRトRレRレRミRミRP4レ
300 PRINT TAB(10);T;CHR$($E2);"デス"
320 PRINT :PRINT "モウイチド チョウセン シマスカ? YES OR NO";
330 INPUT R$:IF R$="YES" THEN GOTO 20
340 END
```

●演算子と式

算術演算子

表1

演算	演算子	例	意味	演算順序
べき乗	^	A ^ B	AをB乗する。	1
かけ算	*	A * B	AにBをかける。	2
わり算	／	A ／ B	AをBでわる。	2
たし算	＋	A ＋ B	AにBをたす。	3
ひき算	－	A － B	AからBをひく。	3

例38 ^（べき乗記号）は－（マイナスキー）の右どなりです。

```
10 PRINT 2^3；2^（－3）················負のべき乗は（ ）でくくる。
20 PRINT 2^（1／2）；SQR（2）······分数のべき乗も（ ）でくくる。
30 END
```

関係演算子

表2

演算子	例	意味	備考
＝	A＝B	AとBとは等しい。	
＜	A＜B	AはBよりも小さい。	
＞	A＞B	AはBよりも大きい。	
＜＝	A＜＝B	AはBと等しいか，Bよりも小さい。	＝＜ ｝は使えない。
＞＝	A＞＝B	AはBと等しいか，Bよりも大きい。	＝＞ ｝は使えない。
＜＞	A＜＞B	AとBとは等しくない。	＞＜ ｝は使えない。

演算の順序

表3

1．(　　)の計算
2．算術関数
3．べき乗
4．かけ算，わり算
5．たし算，ひき算
6．関係演算（＝，＜，＞，＜＝，＞＝，＜＞）
同じ順序のときは左から右へ。

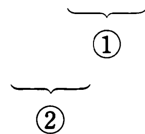
例39 演算の順序をたしかめてください。

```
10 LET A＝2：LET B＝3：LET C＝4：LET D＝5
20 PRINT A＋B*C＋D
30 PRINT （A＋B）*（C＋D）
40 PRINT B－C／B^A
50 PRINT （（B－C）／B）^A
60 END
```

式中の関係演算子

1．式の中に関係演算子（=，＜，＞，＜=，＞=，＜＞）を書くことができる。
2．関係が成立すれば“1”，不成立なら“0”となる。
3．関係演算の順位は，四則演算の後。前ページ「演算の順序」の表3を参照。

A＝B＞5＊C　の式でC＝2なら　　A＝B＞10



B＝15なら
B＞10は成立
従ってA＝1

B＝6なら
B＞10は不成立
従ってA＝0

例40

```
10  LET  B=1
20  LET  A=(B>1)+(B>3)+(B>5) ………複数個の関係演算子を
30  PRINT  B，A                        式の中に入れるときは
40  LET  B=B+1                         （ ）でくくる。
50  IF  B<=6  GOTO  20 ………………Bが6以下なら20行めに跳ぶ。
60  END
```

例41

```
10  INPUT  A:LET  A=INT(A) ………数値を入力する。その整数
20  LET  B=A/2:LET  C=INT(B) …その半分の，整数部。
30  LET  D=B-C ……………………………Bの小数部を計算
40  LET  E=(D=0)+(A>=10) …………Dが0（偶数）かAが10以
50  IF  E>0  THEN  GOTO  10        上ならEは正。
60  MUSIC  トミソ
70  GOTO10
```

式

式とは〈単純数値変数〉，〈添字付数値変数〉，〈定数〉，〈算術関数〉，数値型に変換する〈文字取扱い関数〉，〈組込み定数〉，〈組込み変数〉を演算子で結び，組み合わせたもの

※文字変数で　A＄＝B＄＋C＄＋〝ABC〟とか，IF　A＄（I）＞B＄（I＋1）といった式に似た記述があるが，これらは厳密な意味では式ではなく，文字列の結合と比較を意味するだけです。

3.3 画面エディター（1文字エディター）

画面エディターは，画面上に表示されているカーソルをキーボードからの操作によって全表示画面内にわたって移動させ，プログラムリストを1文字単位で修正できる編集機能です。

1．操作方法

修正したいリストを画面に表示させた後，カーソルを下表の操作によって目的の文字位置まで移動させ，削除には DEL キーを，挿入には挿入したい文字のキーをおします。

修正が完了したら必ず RETURN キーをおします。この操作ではじめてメモリー内のプログラムが修正できます。

カーソルの移動操作

移動方向	操　　作	注　意　事　項
右	英記号 をおしたまま → をおす	1．キートップの矢印がカーソルの移動方向を示します。 2．カーソルは上下方向には同じ縦の列しか移動しませんが，水平方向には，自動的に次の行に移動します。
左	英記号 をおしたまま ← をおす	
上	英記号 をおしたまま ↑ をおす	
下	英記号 をおしたまま ↓ をおす	

修正手順

編集内容	操 作 方 法	（例）
挿入	挿入したい文字をキー入力するとカーソルの指している位置にその文字が入り，それ以後の文字が1字ずつ右にずれる。	(前)　20　MUSIC　トレフソラシ（フ↑カーソル） (後)　20　MUSIC　トレミフソラシ（ミ↑挿入，フ↑カーソル）
削除	DEL をおすと，そのカーソル位置の1つ前の文字が消え，1文字つまる。	(前)　20　MUSIC　トトレミフソラシ（ト↑削除したい文字，レ↑カーソル） (後)　20　MUSIC　トレミフソラシ
完了	RETURN キーをおす。	RETURN キーをおすことにより，カーソルが位置している行についての修正が完了します。

2．注意事項

(1)行番号を修正すると，修正前の行番号のステートメントはそのまま残り，修正した行番号のステートメントが新たに加わることになります。

```
 10 LET A=25  ┐        10 LET A=25 ……………… このまま残る。
     ↓        ├→
210 LET A=25  ┘
↑―― 1文字エディターで2を挿入    210 LET A=25 ……………… 修正後の1行が
                                                          新たに加わる。
```

(2)修正完了を示す RETURN キーは，RETURN キーがおされた時点でカーソルが位置している１ＢＡＳＩＣ文に限ってのみ有効です。従って，２つの文にまたがった修正を一回の RETURN キーの入力で完了することはできません。

例

```
10  MUSIC  トレミレ          もとの文
20  MUSIC  フソラシ
        ↓
10  MUSIC  トレミフレ        フを挿入して，カーソルを２０行に移し，
20  MUSIC  フソラレシ        レを挿入して RETURN キーをおした場合，
        ↓
10  MUSIC  トレミレ          １０行めは修正されず、RETURN キーがお
20  MUSIC  フソラレシ        された位置の２０行のみが修正される。
```

(3)プログラムを省略形で入力してＬＩＳＴすると,１ＢＡＳＩＣ文の長さが７９字以上になる場合があります。このような文を再入力しても,１ＢＡＳＩＣ文が７９字以上のままだと入力エラーになりますので７９字以下となるように修正して再入力してください。

(4)１文字単位の修正は，画面上に必要なステートメント以外表示されていないことを確めて行なってください。画面上に図形などを表示し，その中で１文字単位の修正を行なうと，RETURN キーをおしたときに図形もプログラムの一部とみなして入力します。あらかじめＣＬＥＡＲして，ＬＩＳＴを表示し，修正すれば安全です。

例 右図のように表示されているステートメントに１文字単位の修正を加えると画面の右側の＊もプログラムの一部とみなして入力してしまいます。

```
*******************************
*
> LIST 10
  10 LET A=B+C
*
```

(5)カーソル移動をともなわない文字列を入力したときは RETURN キーをおすまでの文字列を入力します。

例

```
                10  REM  MUSIC  ウタ     ①１０行めを修正し，RETURN
カーソル───→_  20  MUSIC  トレミ        ②２０行の前にカーソルがくる。
                      ↓
        15  MU  P2V3_  20MUSIC  トレミ
               ↑───────③カーソル移動操作（→←↑↓）をせずに挿入 RETURN
移動してきた                この場合，１５行が追加されてＯＫですが。
カーソル──→_  20  MUSIC  トレミ  ④カーソルを移動操作によって２０行の
                                  先頭に移動した場合，挿入できない。
                                 ⑤ＳＹＮＴＡＸ　ＥＲＲＯＲとなる。
```

ヒント　プログラムの編集

プログラム作りにあたっては，この画面エディターの他に，ＳＥＱ，ＲＥＳＥＱ，ＤＥＬ，ＭＥＲＧＥを利用すると能率がグーンとあがります。
さらにプリンターＭＰ－１０４１をお持ちの方はＬＩＳＴ＃でプログラムリストを打ち出せばもっと便利になります。

3.4 自動スクローリング

自動スクローリングとは画面が文字や記号で一杯になると自動的に画面が上に移動することをいいます。次の操作方法により上に移動する速度をキーボードのキー入力により可変することができます。

操作方法

画面が自動スクローリングしているとき，次に示すキーをおすと速度が可変できます。なお，キーをおしていない状態（この状態を無入力状態とします。）では自動スクローリングが一番速くなります。

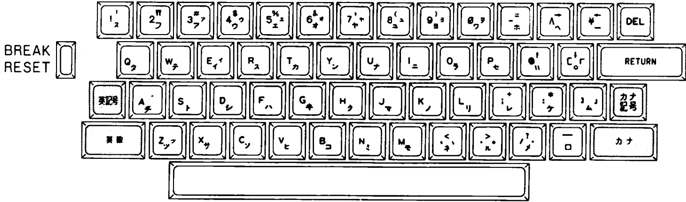
速　度	操　　　作
高　速	無入力
中高速	[英 数] をおす
中　速	[英記号] をおす
中低速	[カナ記号] をおす
低　速	[カ ナ] をおす

注１ [カナ記号] をおしたまま，スクローリングを止めようと [BREAK RESET] をおすと，リセット状態となりプログラムが消失しますので十分ご注意下さい。必ず [カナ記号] キーをはなしてから [BREAK RESET] をおすようにしてください。

例 プログラムを入力し [R][U][N][RETURN] としてプログラムを実行させてから [英 数][英記号][カナ記号][カ ナ] を順におしてみてください。

```
10 CLEAR
20 FOR X=1 TO 1000
30 PRINT X
40 NEXT X
50 END
```

なお，自動スクローリングの速度可変は上の例のようにプログラムを実行しているだけでなく，次頁のＬＩＳＴ出力時にも有効です。



3.5 コマンド

コマンドは，行番号をつけず入力し，コンピューターに各種の動作指示を与える役割をします。

[R][U][N]　　省略形 [R]　　[R][U][N]〈行番号n〉 省略形 [R]〈行番号n〉

プログラムの先頭から実行する。	行番号nからプログラムを実行する。

[N][E][W]

> プログラムを消去し画面もクリアする。

[L][I][S][T]　省略形 [L]

> プログラムのリストを画面に出す。(プログラムを省略形でキー入力しても全てフルスペルでリストされます。)

1．LIST [RETURN]　(L [RETURN])　リスト全てを連続して画面に出す。リストが一画面以上のときは，リストをせりあげながら表示する。途中で止めるときは [BREAK RESET] を押す。

2．LIST〈行番号n〉[RETURN]　(L〈行番号n〉[RETURN])　行番号nの一行のみを画面に出力する。

3．LIST〈行番号n〉，[RETURN]　(L〈行番号n〉，[RETURN])　行番号nから最終行までを画面に表示する。

4．LIST，〈行番号n〉[RETURN]　(L，〈行番号n〉[RETURN])　行番号の最初から行番号nまでを画面に表示する。

5．LIST〈行番号n〉，〈行番号m〉[RETURN]　行番号nから行番号mまでを画面に表示する。

例．LIST（またはL）　10，100 [RETURN]
　　10行から100行までのリストを表示する

[L][I][S][T]＃ 省略形 [L]＃

> プログラムのリストをプリンターに出す。

LIST＃，L＃を1つの独立したコマンドとして取り扱う。LISTと＃との間，Lと＃の間にスペースをつけるとエラーになる。

1．LIST＃ [RETURN]　(L＃ [RETURN])　リストを全て連続してプリンターに出力する。途出で止めたいときは [BREAK RESET] を押す。

2．LIST＃　〈行番号n〉[RETURN]　(L＃　〈行番号n〉[RETURN])　行番号nの一行のみをプリンターに出力する。

3．LIST＃　〈行番号n〉，[RETURN]　(L＃　〈行番号n〉，[RETURN])　行番号nから最終行までをプリンターに出力する。

4．LIST＃，〈行番号n〉[RETURN]　(L＃　，〈行番号n〉[RETURN])　行番号の最初から行番号nまでをプリンターに出力する。

5．ＬＩＳＴ＃　〈行番号ｎ〉，〈行番号ｍ〉 RETURN　行番号ｎから行番号ｍまでを
（Ｌ＃　〈行番号ｎ〉，〈行番号ｍ〉 RETURN ）　プリンターに出力する。

注１．プリンターが接続されていないとき，またはプリンターが動作可能な状態でないとき，ＬＩＳＴコマンドを実行すると約３秒後に PRINTER NOT READY BREAK が表示されます。
注２．本命令での印字モードは，80桁／行です。他のモードでのご使用は，107ページプリンターの使い方の項をご覧ください。

SEQ

行番号を自動的に出力する。

1．ＳＥＱ RETURN　行番号１０から１０きざみで行番号を自動的に出力する。
2．ＳＥＱ　ｎ RETURN　行番号ｎから１０きざみで行番号を自動的に出力する。
　またはＳＥＱ　ｎ， RETURN
3．ＳＥＱ　ｎ，ｋ RETURN　行番号ｎからｋきざみで行番号を自動的に出力する。
4．ＳＥＱ　，ｋ RETURN　行番号１０からｋきざみで行番号を自動的に出力する。

ＳＥＱの解除
一たんＳＥＱコマンドを入力し，順次プログラムを作成していくと RETURN をおすたびに次の行番号が出力されますが，これを解除するには行番号が出力された直後に RETURN をおせば ＞_ マークが出て解除できます。

ＳＥＱと画面エディター
ＳＥＱでプログラムを作成中,その行,または前の行の誤りに気がついて画面エディターでカーソルを移動し，修正を行なって元の行にもどると，SEQした行番号が１きざみ分だけ増えます。 ＳＥＱの途中で画面エディターを使用したいときは，一たんＳＥＱを解除し,画面エディターで修正後再びＳＥＱコマンドで続行することをおすすめします。

RESEQ

プログラムの先頭から終りまでの行番号を自動的につけかえる。
（このとき，ステートメントの中にある行番号，例えばＧＯＴＯ２００とか，ＧＯＳＵＢ２０も自動的につけかえられる。）

1．ＲＥＳＥＱ RETURN　リストの先頭にある行番号を１０に定め，１０きざみでリストの終りまでの行番号をつけかえる。
2．ＲＥＳＥＱ　ｎ RETURN　リストの先頭にある行番号をｎに定め，１０きざみでリストの終りまでの行番号をつけかえる。
　又はＲＥＳＥＱ　ｎ， RETURN
3．ＲＥＳＥＱ　ｎ，ｋ RETURN　リストの先頭にある行番号をｎに定め，ｋきざみでリストの終りまでの行番号をつけかえる。
4．ＲＥＳＥＱ　，ｋ RETURN　リストの先頭にある行番号を１０に定め，ｋきざみでリストの終りまでの行番号をつけかえる。

注１．未定義の行番号があるときは，その行番号をそのままにしてＲＥＳＥＱを続行する。例次頁

例

```
10 INPUT A
20 IF A=1000 GOTO 500
30 ------------
```

└─RESEQするとこの行番号はそのまま500

注2．ON…GOTO，ON…GOSUBで，跳び先番号として記述されている行番号が存在しないと，その行番号以後に書かれている行番号は変更されずRESEQを続行することがあります。

[D][E][L]

指定する行番号を消去する。（部分消去）

1．(DEL)※n [RETURN]	行番号nを1行のみ消去する。（行番号が存在しない場合は，SYNTAX ERROR と表示される。） ※1行消去のときはDELも省略できる。
2．DEL n, m [RETURN]	行番号nからmまでのリストを消去する。
3．DEL n, [RETURN]	行番号nからリストの終りまでを消去する。
4．DEL , m [RETURN]	行番号の先頭から行番号mまでのリストを消去する。

注1．非常に長いプログラムをDELコマンドで消去するとかなりの処理時間がかかります。このとき [BREAK RESET] をおしてもBREAKがかかりません。DELコマンドの実行が終り，>_ マークが出るまでしばらく待ってください。

[S][I][Z][E]　省略形 [S]

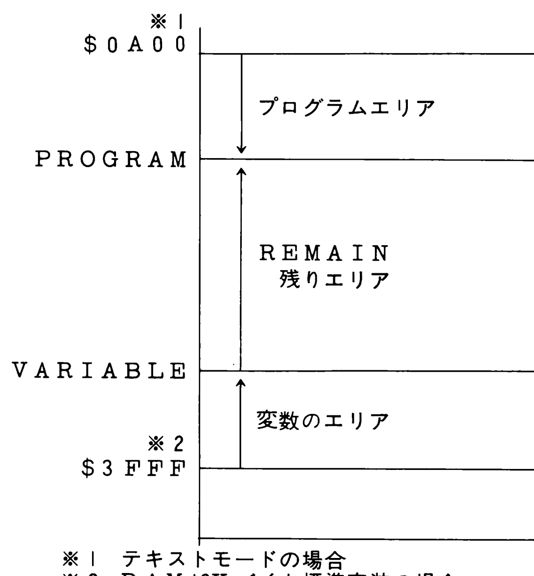
プログラムの大きさと，残りのメモリーの大きさを表示する。

BASICのプログラムはテキストモードでは$0A00番地から格納され，プログラムで使われる変数は，プログラムの実行の順番にしたがってメモリーの最終番地，標準実装で$3FFF番地から格納されます。

SIZE [RETURN] によって

PROGRAM ＝$○○○○ ⎫ 16進数
VARIABLE＝$○○○○ ⎭
REMAIN ＝△△△△△ 10進数

が表示されます。



※1　テキストモードの場合
※2　RAM16Kバイト標準実装の場合

[C][O][N][T][I][N][U][E]　　省略形 [C][O][N][T] および [C]

1．ＳＴＯＰステートメントで停止したプログラムを，ＳＴＯＰの次のステートメントから実行する。
2．ＢＲＥＡＫで停止したプログラムを，ＢＲＥＡＫした次のステートメントから実行する。
例外　1）ＢＲＥＡＫがＩＮＰＵＴ，ＭＵＳＩＣステートメント中で行なわれた場合，ＣＯＮＴＩＮＵＥはできない。
　　　2）ＢＲＥＡＫがマルチステートメントで行なわれた場合，：で結合されたステートメントかあるいは次のステートメントから実行する。

[M][O][N][I][T][O][R]　　省略形 [M][O][N]

モニターにジャンプします。このとき，ＢＡＳＩＣのプログラムは破壊されることがあります。詳しくは「モニターの使い方」を参照してください。

[S][A][V][E]

プログラムをカセットテープに記録します。詳細は９７ページ。

[L][O][A][D]

プログラムをカセットテープから本体内メモリーに再生格納します。詳細９９ページ。

[M][E][R][G][E]

メモリー内に格納されているプログラムに，カセットテープから読み込んだ別のプログラムを結合します。詳細は９９ページ。

[V][E][R][I][F][Y]

ベーシックマスターのメモリー内にあるプログラムと，カセットテープに記録してあるプログラムとの内容を比較確認します。詳細は９８ページ。

3.6 標準的なステートメント/●LETステートメント

キーワードＬＥＴで始まるステートメントは，変数（添字付変数も含む）に式の値を代入させる（割当てる）役目をします。

> ＬＥＴ　〈数値変数〉または〈添字付数値変数〉＝〈式〉
> ＬＥＴ　〈文字変数〉または〈添字付文字変数〉＝〝文字列〞
>
> キーワードＬＥＴは無条件に省略できる。

ここで〈式〉とは，３５ページで定義したものを言います。

１．〈定数〉　例　ＬＥＴ　Ａ＝２７
２．〈変数〉　例　ＬＥＴ　Ｂ＝Ｃ
３．〈式〉　例　ＬＥＴ　Ｄ＝３＋４＊Ｈ－４
４．〈算術関数〉　例　ＬＥＴ　Ｅ＝ＲＮＤ（６）
５．〈〝文字列〞〉，〈文字取り扱い関数〉
例　ＬＥＴ　Ａ＄＝〝ＡＢＣ〞＋ＣＨＲ＄（＄４５）

例 42

```
10 LET A=27 ……………… Aに定数27を代入する。
20 LET B=A*3+18 ……… Bに式，A×3+18を代入する。
30 PRINT B ……………… Bをプリント。
40 END …………………… プログラム終了。
```

例 43　文字列を文字変数に代入させる場合の例です。

```
10 LET W0$="ワタシ ノ " ……… 文字列は〝 〞でくくって代入。
20 LET W1$="ナマエ ハ " ……… 文字変数には必ず$（ドル記号）を。
30 W3$="デ ス。" …………… LETは省略しても全くさしつかえない。
40 INPUT W2$ ……………… ?が出たら名前をキー入力します。
50 CLEAR …………………… 画面をクリアーして左上からプリント。
60 PRINT W0$;W1$;W2$;W3$
70 END
```

例 44　階乗を求めるプログラム例です。

```
10 REM *** Nノカイジ ョウ***
20 INPUT "N=",N ……………… Nは正の整数で入力してください。
30 LET M=N:LET L=M ………… Mは80行でNをプリントするため。
40 IF N=1 GOTO 80
50 LET N=N-1
60 LET L=L*N ……………… このときのNは50行のN-1。
70 IF N>1 GOTO 50
80 PRINT M;"!=";L ……… 入力値N!=○○と表示させるため。
90 END
```

●PRINTステートメント

画面に変数や定数などの数値や，文字を出力するステートメントです。

```
PRINT　&lt;リスト&gt;

　PR　&lt;リスト&gt;　または　?　&lt;リスト&gt;　と省略してもよい。

&lt;リスト&gt;には以下のものを記述できる。

1．&lt;変数&gt;  { &lt;単純数値変数&gt;      例） PRINT X
             { &lt;添字付数値変数&gt;    例） PRINT Y（1，2）
             { &lt;単純文字変数&gt;      例） PRINT B$
             { &lt;添字付文字変数&gt;    例） PRINT Z$（2）

2．&lt;式&gt;                            例） PRINT A+2*C/6

3．&lt;定数&gt;  { &lt;数値&gt;              例） PRINT 123.4
             { &lt;"文字列"&gt;          例） PRINT "イロハニホヘ"

4．&lt;関数&gt;  { &lt;算術関数&gt;          例） PRINT SIN（X）
             { &lt;文字取り扱い関数&gt;  例） PRINT LEN（P1$）
             { &lt;特殊関数&gt;          例） PRINT PEEK（$A）
             { &lt;PRINT文用関数&gt;     例） PRINT TAB（A*3）

5．&lt;その他&gt;{ &lt;組み込み変数&gt;      例） PRINT TIME
             { &lt;組み込み定数&gt;      例） PRINT PAI

これら1～5までを区切り記号※を使って並べて書くことができる。
（PRINTも含めて最大79字まで。）　　※次頁参照。
```

例 45　ダイレクト実行のとき，PRINT文をよく使います。

```
?30/2.54 ……………… センチをインチに換算します。
```

例 46　区切り記号によって画面に表示される様子が大きく変化します。実際にキー入力してその違いを確めてください。次頁に表にしてまとめてあります。

```
10 A=1.5324:B=0.1234:C=-3.3
20 PRINT A
30 PRINT B
40 PRINT C
50 PRINT A,  ……… }
60 PRINT B,       } PRINT A，B，Cと同じ意味です。
70 PRINT C   ……… }
80 PRINT A;B;C
90 END
```

表示桁とは，表示する数値の桁に，±の符号桁，小数点および＄記号を加えた桁をいう。(例えば　－３．１３１７　の５桁の数は７桁の表示桁となる。)

	桁指定	最大表示桁	区切記号	内　容	条件および例外項目	例　A＝1.2343　B＝3.14159265　C＝－３　D＝1.5×10^{-10}
※※ 10進数	％ｎ；※ (％ｎの直後は必ず；で区切る。)	ｎ	，	ｎ＋１をゾーンに設定し，ｎ桁の表示桁で左から順次ツメて表示される。(このときｎ＋１桁からｎ桁へは四捨五入されて表示される。)	①３≦ｎ≦１１であること，ｎがこれ以外のときはｎが無指定と同じ。 ②指数形の数値表示はＥ±ＸＸだけゾーンからはみでる。したがって２つのゾーンを使うことがある。※※※ ③同一ＰＲＩＮＴ文中では％で新たに表示桁を定義しないかぎりｎ桁の指定が有効。	例　ＰＲＩＮＴ　％６；Ａ，Ｂ，Ｃ ␣１.２３４␣␣３.１４２␣－３␣␣␣␣␣ |←７→|←７→|←７→| 例　ＰＲＩＮＴ　％６；Ｄ，Ｂ ␣１.５Ｅ－１０␣␣␣␣␣␣␣␣３.１４２␣ |←７→|←７→|←７→|
		ｎ	；	ゾーンは設定されず前の表示桁に続けて表示される。		例　ＰＲＩＮＴ　％６；Ａ；Ｂ；Ｃ；Ｄ ␣１.２３４␣３.１４２－３␣１.５Ｅ－１０ |←６→|←６→|←２→|←８→|
	無指定 (％ｎ；を省略した場合)	１１	，	８桁のゾーンが設定され，ｎ＝１１桁の表示桁で左から順次ツメて表示される。	①８桁以上の表示桁および８文字以上の文字列の表示桁のときは２つのゾーンを使い１６桁で表示される。	例　ＰＲＩＮＴ　Ａ，Ｂ，Ｃ ␣１.２３４３␣␣３.１４１５９２６５␣␣␣␣␣－３␣␣␣␣␣␣ |←８→|←８→|←８→|←８→|
		１１	；	ゾーンは設定されず，前の表示桁に続けて表示される。		例　ＰＲＩＮＴ　Ａ；Ｂ；Ｃ；Ｄ ␣１.２３４３␣３.１４１５９２６５－３␣１.５Ｅ－１０ |←７→|←１１→|←２→|←８→|
16進数	％ｎで指定できない。	５	，	１４桁のゾーンで左ツメで表示される。		例　ＰＲＩＮＴ　ＨＥＸ（＄ＦＦ），ＨＥＸ（＄１２ＡＥ） ＄００ＦＦ␣␣␣␣␣␣␣␣␣＄１２ＡＥ␣␣␣␣␣␣␣␣␣ |←１４→|←１４→|
		５	；	６桁のゾーンで表示される。		例　ＰＲＩＮＴ　ＨＥＸ（＄ＦＦ）；ＨＥＸ（＄１２ＡＥ） ＄００ＦＦ␣＄１２ＡＥ␣ |←６→|←６→|

※　，で区切ると左はしより８桁のゾーンをとばし，次のゾーンから％ｎの指定を行なう。

※※　０．ＸＸＸは　．ＸＸＸまた，Ｘ．ＸＸ０は　Ｘ．ＸＸと０を省略して表示される。

※※※　指数形の数値表示のときは仮数部のみの表示桁指定が行われ，指数部には指定が適用されません。

ＰＲＩＮＴ文用関数

関　数	機　能	例
ＴＡＢ（〈式〉）※	ＰＲＩＮＴステートメントの実行開始時のカーソル位置を基準として〈式〉の値だけカーソルを右へ移動する。(空白は出力しない。)	ＰＲＩＮＴ　ＴＡＢ（３）；Ｂ ␣␣␣－１００ TAB(3)
ＨＥＸ（〈式〉）※※	〈式〉の値を１６進数４桁で出力する。	ＰＲＩＮＴ　ＨＥＸ（２５６） ＄０１００␣

※〈式〉の値の有効範囲は０～２５５です。〈式〉＝０のときのカーソル位置はＰＲＩＮＴステートメントの実行開始時のカーソル位置となります。
※※〈式〉の値の範囲は－６５５３５～６５５３５です。ただし，〈式〉の値が負の数（２の補数表現）のときには６５５３６－(〈式〉)の値を１６進数４桁で出力します。

書式制御（ＣＨＲ＄関数）

指　定	内　容
ＰＲＩＮＴ　ＣＨＲ＄(＄０１)；	文字図形のコード表中＄００～＄０Ｆの部分を出力するモードを設定。
ＰＲＩＮＴ　ＣＨＲ＄(＄０２)；	画面をスクロールアップ。
ＰＲＩＮＴ　ＣＨＲ＄(＄０３)；	画面をスクロールダウン。
ＰＲＩＮＴ　ＣＨＲ＄(＄０４)；	何もしない。
ＰＲＩＮＴ　ＣＨＲ＄(＄０５)；	カーソルをブリンク（点めつ）表示。
ＰＲＩＮＴ　ＣＨＲ＄(＄０６)；	カーソルを消去。（但しキー入力時１度だけ表示。）
ＰＲＩＮＴ　ＣＨＲ＄(＄０７)；	ベル音（ピーという音）の発生。
ＰＲＩＮＴ　ＣＨＲ＄(＄０８)；	カーソルを１文字分左へ移動。
ＰＲＩＮＴ　ＣＨＲ＄(＄０９)；	カーソルを１文字分右へ移動。
ＰＲＩＮＴ　ＣＨＲ＄(＄０Ａ)；	カーソルを１文字分下へ移動。
ＰＲＩＮＴ　ＣＨＲ＄(＄０Ｂ)；	カーソルを１文字分上へ移動。
ＰＲＩＮＴ　ＣＨＲ＄(＄０Ｃ)；	画面を消去しカーソルをホームへ移動。
ＰＲＩＮＴ　ＣＨＲ＄(＄０Ｄ)；	カーソルを次の行の先頭へ移動。
ＰＲＩＮＴ　ＣＨＲ＄(＄０Ｅ)；	白地に黒文字の画面にする。
ＰＲＩＮＴ　ＣＨＲ＄(＄０Ｆ)；	黒地に白文字の画面にする。
ＰＲＩＮＴ　ＣＨＲ＄(＄７Ｆ)；	カーソルを左へ１つ移動してスペースを出力。（DELキーと同じ役割）

※ＣＨＲ＄に続くカッコ内の数は，上の例のように１６進数でなく，１０進数でもかまいません。１６進，１０進いずれの場合も０は省略できます。
※※末尾には必ず；（セミコロン）を入れて下さい。

文字，図形の表示コード表

X：上位４ビット，Y：下位４ビット

Y＼X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		α		0	@	P		p	┼			ー	タ	ミ	時	
1		β	!	1	A	Q	a	q			。	ア	チ	ム	分	
2		γ	"	2	B	R	b	r			「	イ	ツ	メ	秒	
3		η	#	3	C	S	c	s			」	ウ	テ	モ	年	
4		θ	$	4	D	T	d	t		○	、	エ	ト	ヤ	月	
5		λ	%	5	E	U	e	u	─	●	・	オ	ナ	ユ	日	
6		μ	&	6	F	V	f	v	│	♣	ヲ	カ	ニ	ヨ	火	
7		π	'	7	G	W	g	w	┼	♦	ァ	キ	ヌ	ラ	水	
8		τ	(	8	H	X	h	x	┤	♥	ィ	ク	ネ	リ	木	
9		Φ	)	9	I	Y	i	y	┬	♠	ゥ	ケ	ノ	ル	金	
A		Ψ	*	:	J	Z	j	z	├		ェ	コ	ハ	レ	土	
B		ω	+	;	K	[	k	←	┴	♪	ォ	サ	ヒ	ロ	人	
C		Σ	,	<	L	¥	l	→			ャ	シ	フ	ワ	点	／
D		Ω	-	=	M	]	m	↓			ュ	ス	ヘ	ン	回	＼
E		χ	.	>	N	^	n	↑			ョ	セ	ホ	゛	文	
F		÷	/	?	O	_	o				ッ	ソ	マ	゜	名	

↑※※

※ＰＲＩＮＴ　ＣＨＲ＄(＄ＸＹ) の形で上の表の１種類を指定します。( ) の中は，１０進数，１６進数，定数，式が使えます。

(１バイトの上位４ビット：Ｘ，下位４ビット：Ｙ　で１６進数０～ＦがＸ，Ｙに入ります。)

※※＄００（Ｘ＝Ｙ＝０）～＄０Ｆ（Ｘ＝０，Ｙ＝Ｆ）の部分の１列の図形は，４６ページ「書式制御」の表の，ＰＲＩＮＴ　ＣＨＲ＄（＄０１）で出力モードを設定したあと，続けて指定します。(但し，出力モードの設定は次の１文字についてのみ有効です。) [例：▞の表示………ＰＲＩＮＴ　ＣＨＲ＄（＄０１）；ＣＨＲ＄（＄０Ｂ）]
ただし，ＣＨＲ＄（＄０４）は，＄０４を区切り記号としてプログラムで使用しているため出力することはできません。

例47　１行は７９字までＯＫ，相当のことが書けます。

```
10  FOR  I=0  TO  $F:?CHR$(1);CHR$(I);:
    NEXT  I:END
```

※※※上の表の図形は，７４ページの **例93** のプログラムで実際に画面に表示してみることができます。

▒の部分はＪＩＳで規定されているコードと一致しています。

例48　ＰＲＩＮＴステートメントは他の例で示していますのでここではプリント関数の例を作ってみます。まずＴＡＢ関数でグラフを描きます。

```
 5 CLR
10 FOR Y=0 TO 18
20 X=27-(Y-9)^2/3 ......... X=27-1/3(Y-9)^2
30 PRINT TAB(X);"*"          Xに上式が入る。
40 NEXT Y
50 END
```

例49　ＣＨＲ＄関数は，４７ページのような図形を書くプログラムだけではなく，カーソル制御等の書式制御の関数となります。このプログラムは枠の中の◆マークが左右に移動し，枠にぶつかるとベル音を発します。このプログラムが理解できたら次に上下に移動させる，さらに二次元の移動と発展してください。

```
 10 CLR
 20 FOR Y=0 TO 20 .......................
 30 IF (Y=0)+(Y=20)=1 THEN A=1:GOTO 50
 40 A=31
 50 FOR X=0 TO 31 STEP A
 60 CURSOR=X,Y                        外枠を描くプログラム
 70 PR CHR$($60);                     です。
 80 NEXT X
 90 NEXT Y ..............................
100 CURSOR=15,10
110 FOR D=1 TO 2 ............ 往きと戻りの条件を変えます。
120 PR CHR$($97); ........... ◆マーク。何でもよい。
130 FOR G=0 TO 10 ...........}
140 NEXT G ..................} 移動の速さ
150 PR CHR$($08); ........... カーソルを左に戻す。($08=8)
160 PR CHR$($20);:Q$=!$ ..... 戻して◆を消去。
170 IF D=1 GO 210 ........... D=1は右への動き
180 PR CHR$($08); ...........}
190 PR CHR$($08);:Q$=!$ .....} 左への動きは2つ戻す。
200 IF Q$=CHR$($60) PR Q$;:GOTO 230
210 IF Q$=CHR$($60) PR CHR$($08);:GOTO 230
220 GOTO 120
230 PR CHR$($07); ........... ベル音発生。
240 NEXT D
250 GOTO 110 ................ 無限ループ。
300 END
```

●ＤＩＭステートメント

ＤＩＭステートメントは，プログラムで添字付変数（配列）を使用したいとき，あらかじめ，プログラムの先頭で配列の大きさを宣言するステートメントです。

```
※
１次元   ＤＩＭ　Ａ（整数），Ｂ（整数），………………………
※
２次元   ＤＩＭ　Ａ（整数，整数），Ｂ（整数，整数）………………………

配列名 ─────────── コンマで区切って複数個を定義できる。
                    配列の大きさを指定する。
```

１．上記宣言はプログラムの先頭（プログラム中で配列が使われる以前に）で行なわなければならない。
2．配列名はＡ～Ｚ，Ａ０～Ｚ９までの２８６種の英数文字が使える。
3．一度配列として宣言した文字は，別の変数として（添字のない単なる変数としても）使えない。ただし，数値変数文字変数とは互に独立して使える。
4．数値変数は１次元，２次元の配列を，文字変数は１次元の配列を宣言できる。
5．一度宣言すれば，プログラム中で添字付変数として使える。
6．このステートメントで宣言可能な添字付変数（配列）の（　）の中は，数値変数のとき１～８５００，文字変数のとき１～２５５です。

※27ページ参照。

例50　配列の（　）の中は１以上の整数。演算の途中でＡ（０）にならないように式をたててください。

```
１０　ＤＩＭ　Ａ（１３）……………………………配列宣言
２０　ＬＥＴ　Ａ（１）＝2…………………………初期値を設定します。
３０　ＦＯＲ　Ｎ＝２　ＴＯ　１３
４０　ＬＥＴ　Ａ（Ｎ）＝Ａ（Ｎ－１）＊2……配列の関係，2ⁿ乗の数列です。
５０　ＰＲＩＮＴ　Ｎ－１，Ａ（Ｎ－１）
６０　ＮＥＸＴ　Ｎ
９９　ＥＮＤ
```

例51　文字列の配列は１次元だけ，３２字までの文字列を代入できる。１０個の配列を始めに書き込み、つぎに番号を入れるとその配列を画面に表示する例です。

```
１０　ＤＩＭ　Ａ＄（１０）………………………………配列宣言
２０　ＦＯＲ　Ｂ＝１　ＴＯ　１０
３０　ＩＮＰＵＴ　〝ＮＡＭＥ〟，Ａ＄（Ｂ）………３２字以内の文字列入力
４０　ＮＥＸＴ　Ｂ
５０　ＩＮＰＵＴ　〝ＮＵＭＢＥＲ〟，Ｃ……………１から１０までの数字を入力
６０　ＩＦ　（Ｃ＝０）＋（Ｃ＞１０）　　　　　ＯＲ論理，Ｃ＝０またはＣが１０を
　　　ＴＨＥＮ　ＧＯＴＯ　５０………………………超えるときは５０行に戻る
７０　ＰＲＩＮＴ　Ａ＄（Ｃ）…………………………30行でＣ番めに入力した文字を出力
９０　ＧＯＴＯ　５０…………………………………無限ループ。止めるときは、ＢＲＥＡＫキーをおす
１００　ＥＮＤ
```

●REMステートメント

プログラムの名前や処理内容をわかり易くするため，コメントを書き入れるステートメントです。

> REM 〈コメント〉
>
> 1．プログラム中のどこの行に書いてもよい。
> 2．実行するときは，REMを無視する。
> 3．REM 〈コメント〉につづくマルチステートメントは許されない。（マルチステートメントも〈コメント〉とみなされる。）

例52 Kg←→Lb（ポンド）の換算プログラムです。REM文には何を書いてもかまいません。

```
 10 REM ＊＊＊ カンザ ン ＊＊＊
100 REM ### PRINT ###
110 PRINT "キログ ラムカラポ ンド デ スカ"
120 PRINT "ポ ンド カラキログ ラムデ スカ"
200 REM ### センタク ###
210 PRINT "K OR P?"
220 INPUT A$:REM KマタハPヲ ニュウリョク
230 IF A$="K" GOTO 400
240 IF A$<>"P" GOTO 500
300 REM %%% ポ ンド ノケイサン %%%
310 PRINT"ナンポ ンド デ スカ";
320 INPUT P
330 LET K=P＊0.45359237
340 PRINT
350 PRINT P;"ポ ンド ハ ";%6;K;" キログ ラムデ ス"
360 GOTO 200
400 REM ¥¥¥ キログ ラムノケイサン ¥¥¥
410 PRINT "ナンキログ ラムデ スカ";
420 INPUT K
430 LET P=K＊2.20462260
440 PRINT
450 PRINT K;"キログ ラムハ";%6;P;" ポ ンド デ ス"
460 GOTO 200
500 END
```

●ＩＦステートメント

判断を行なうステートメントです。ある条件をテストし，条件が成立すれば，条件文のつぎのステートメントを実行し，条件が成立しなければ，次の行番号のステートメントを実行します。

ＩＦ　〈条件〉（ＴＨＥＮ）〈ステートメント〉（ＴＨＥＮは省略できる。）※

――この条件が成り立つなら，――このステートメントを実行。

――この条件が成り立たないときは次の行番号のステートメントを実行。

〈条件〉とは，判断のために，それが成り立つか否かを確めるための〈式〉を言う。

例	ＩＦ　Ａ＞３	Ａが３よりも大きかったら………条件成立
	ＩＦ　Ａ＝２	Ａが２と等しかったら……………条件成立
	ＩＦ　Ａ＜Ｂ＋４	ＡがＢ＋４よりも小さかったら…条件成立
	ＩＦ　Ａ＜＞９９	Ａが９９に等しくなかったら……条件成立

└――判断のための〈式〉，すなわち〈条件〉

ＩＦ　〈条件〉　ＴＨＥＮの後がＧＯＴＯ〈行番号〉のときは　ＩＦ　〈条件〉　ＴＨＥＮ　〈行番号〉のようにＧＯＴＯを省略することもできる。※※

※ＴＨＥＮにつづくステートメントがＰＲＩＮＴ，ＣＵＲＳＯＲの省略形，？，！のときはＴＨＥＮは省略できない。　ＩＦ　〈条件〉　ＴＨＥＮ　！＝１，１０：？Ａ

※※ＴＨＥＮとＧＯＴＯの両方を省略することはできない。

例53　ＲＮＤ（Ｘ）を使ったプログラム例です。コンピューターは１から６までの数字を出します。あなたはそれが何であるかを当ててください。このプログラムは，おもしろくするためにプログラムを追加していきますのでＮＥＷで消さないようにしてください。

```
10  REM サイコロ ゲーム
15  RANDOMIZE
20  LET X=INT (RND (6))…………………0から5までの数字。
30  LET A=X+1………………………………1を加え1から6にする。
60  INPUT B…………………………………思った数字を入力 [RETURN]
70  IF A=B GOTO 200……………………判断。当り！200行に。
80  GOTO 400…………………………………外れ，400行に。
200 REM アタッタトキ
230 PRINT TAB (12); "オオアタリ"
400 REM ハズレタトキ
420 PRINT "サイコロハ" ;A; "デス"
999 END
```

６分の１の確率ですから，なかなか当らないかもしれません。このプログラムは１回きりで終りです。毎回，ＲＵＮとしてプログラムをスタートしてください。

つぎにこのプログラムに〝ＮＯ〟と言うまで何回でもくり返すプログラムを追加します。

例54　〝NO〞とキー入力すると実行終了のプログラム。

```
500　REM　＊＊　NO　デ　ヤメル　＊＊
510　PRINT
520　PRINT　〝マダ　ヤルキ？　ヤメルナラ　NO　トイッテ
　　　クダサイ〞
530　INPUT　C$
540　IF　C$＝〝NO〞　GOTO　999
550　CLEAR
560　GOTO　15
 50　PRINT　“アナタノスキナ　バ　ンゴ　ウヲド　ウゾ　”；
```

これで，続けて番号をキー入力するときは“NO”以外のキー，たとえば RETURN をおせば何回でもくり返します。しかし，これだけではまだものたりません。もう少し工夫してみましょう。

例55　ついでに，当ったとき音楽が流れてくるプログラムをつくります。200行めからのブロックにMUSICステートメントを追加すればOKです。

```
240　FOR　N＝1　TO　24
250　PRINT　CHR$（$03）；
260　NEXT　N
270　MUSIC　T2MミP3ソ#フソP5UトP3ソミミミ#レミ
　　P5ソP3ミトトミレトP5レP5レP3レレレフミレP5ソP3ミP7ト
280　GOTO　500
410　MUSIC　T3P4トP2トP1トP4トP2#レP1レP2レ
　　P1トP2トP1DシP5ト
```

以上のように，1つのプログラムを発展させながら作っていくことも一つの方法です。このプログラムには，コンピューターと，人とで，点を持ち合ったりするなど，色々なバリエーションが考えられます。工夫して楽しいゲームのプログラムを作ってください。

ヒント　ベーシックマスターに感情表現をさせよう。

```
10　MUSIC　T1V5Q2P5トミソP8Uト
```

合格！　万才!!　といった明るい感じになります。

```
20　MUSIC　T2V5Q1P5トDソBD
　　ミQ3P7Dト
```

不合格！　残念!!　といったややがっかりした感じになります。

プログラムの中にこんなステートメントを入れて楽しいプログラムミングを行なってください。

文字列の比較

ＩＦステートメントの〈条件〉に文字列の比較が用いられた場合，数値の比較とは取り扱いが異なります。

●文字列の比較は，関係演算子をはさんだ２つの文字列を左から順に，各文字のコード（47ページの文字，図形の表示コード表）の大小で，大きさを比較する。

例）　アルファベットのＡとＢとを比較したいとき，コード表によりＡは＄４１，Ｂは＄４２のため，＄４１＜＄４２，従って〝Ａ〟＜〝Ｂ〟となる。

●文字が連続している場合，左から順に判定し，異なった文字のところで判定する。

例）　Ａ＄＝〝サトウイチロウ〟，Ｂ＄＝〝サトウタロウ〟の場合，左から３文字は互いに等しいので，４文字めのイとタを比較する。イ（＄Ｂ２），タ（＄Ｃ０）であるからこの場合，Ａ＄＜Ｂ＄となる。（カナの場合，大体５０音順）

●２つの文字列の長さが異なるとき，短い方の文字列の長さだけ比較する。

例）　Ａ＄＝〝サトウサブロウ〟，Ｂ＄＝〝サトウ〟の場合，短い方の文字列，Ｂ＄のサトウの３字までで比較する。従って　Ａ＄＝Ｂ＄となる。

〈ご注意１〉

●文字列の比較の場合，〈条件〉を（　　）でくくることはできません。

例）　ＩＦ　（Ａ＄＝〝ＹＥＳ〟）　ＴＨＥＮ　１００…ＳＹＮＴＡＸ　ＥＲＲＯＲ

例）たとえば，Ａ＄とＢ＄とを文字列とが等しくかつ長さも等しいという条件式を

ＩＦ　（Ａ＄＝Ｂ＄）＊（ＬＥＮ（Ａ＄）＝ＬＥＮ（Ｂ＄））　ＴＨＥＮ　１００

といったＡＮＤ演算で行うと，ＳＹＮＴＡＸ　ＥＲＲＯＲ　となります。

ＬＥＮ（Ｘ＄）は数値形の関数ですから，文字列の比較と数値の比較とは分けて，ＩＦ文の中に書くことが必要です。このＡＮＤはつぎのように書けます。

ＩＦ　Ａ＄＝Ｂ＄　ＩＦ　ＬＥＮ（Ａ＄）＝ＬＥＮ（Ｂ＄）　ＴＨＥＮ　１００

●文字列の比較を〈式〉の中に書くことはできません。

例）　１０　ＬＥＴ　Ａ＝（Ａ＄＝〝ＹＥＳ〟）…………ＳＹＮＴＡＸ　ＥＲＲＯＲ

〈ご注意２〉

４７ページの文字，図形の表示コード表のうち＄００〜＄０Ｆおよび＄７Ｆは，システムのコントロールコードとして使用しているため，文字比較にこれらのコードは使用できません。

例56 文字列の比較のまとめとして，データをファイルするようなプログラムの例を示します。このプログラムは，一応電話帳を想定して，名前と，名前に対応するデータ（電話番号）とが一対一で対応し，名前，電話番号とも文字列として取り扱います。全体のプログラムは少し長めですが，一つ一つのルーチンはせいぜい１０行程度ですから，コメントと一緒にご覧いただければ理解できるものと思います。英文でプログラムしていますが適時カナ等に直してください。

１．機能概要

a) ＷＲＩＴＥ　ファイルへの書き込み

名前，データを入力すると，名前の文字列の大小を比較し，ファイルの適切な位置に挿入していく。長さも等しい場合は後に入れる。

b) ＤＥＬＥＴＥ　ファイルからの抹消

ファイルをさがし，等しい名前があったら，ファイルから抹消し，その分，ファイルを順次つめる。全く等しい名前があるときは，書き込みと反対に先頭にファイルされているデータを抹消する。

c) ＲＥＡＤ　データ読み出し，表示。

名前を入力すると，〝文字列〟として等しくかつ，長さが入力したもの以上のファイルを表示する。頭文字，姓などによる検索ができる。

例

ＮＡＭＥ	ＤＡＴＡ
Ａ	１２３４
ＡＡ	５６７８
ＡＡＡ	３５２……………
ＢＢＡ	０４５－８８１…

とファイルされているとき，ＡＡで検索すると，ＡＡとＡＡＡを表示する。Ａは表示しない。

d) ＣＬＥＡＲ　全てのファイルを消去する。

間違えて全て消去しないよう２重の入力が必要。

２．ソースプログラム　プリンターでプリントしたものを約７０％に縮少して掲載しています。このため，一行が４０字詰めになっています。

```
  10 DIM N1$(41),D1$(41),R1$(4)
  20 REM *** HARD START ***
  30 LET P=0:LET N1$(1)=CHR$($FF)
  40 LET R1$(1)="WRITE":LET R1$(2)="DEL
ETE"
  50 LET R1$(3)="READ":LET R1$(4)="CLEA
R"
 100 REM *** SOFT START ***
 110 CLEAR :LET CURSOR=3,4:PRINT "****
TELEPHONE LISTS ****"
 120 PRINT :PRINT "  ";P;" DATA ARE ALR
EDY FILED. ":PRINT
 130 PRINT "    WRITE TO FILE..........
1":PRINT
 140 PRINT "    DELETE FROM FILE.......
2":PRINT
 150 PRINT "    READ FILE BY NAME......
3":PRINT
 160 PRINT "    CLEAR ALL FILE.........
4":PRINT
 170 PRINT :PRINT :INPUT "    SELECT NUM
BER.........",K
 180 ON K GOTO 200,500,600,700
```

10：配列宣言，文字列は１つのデータで３３バイト消費します。

20：ＣＬＥＡＲを選んだときだけここにジャンプします。

30～40：Ｐはファイルする人数。ＣＨＲ＄（＄ＦＦ）は，文字列で最も大きいキャラクター，先頭にあらかじめ書きます。常にこの記号はファイルの末尾にある。

100：次の動作のとき，ここにジャンプしてきます。ＢＲＥＡＫで実行を停止したときもここからスタートすると，データをこわさずにプログラムに戻れます。ＲＵＮ　１００はダメです。
　　ＧＯ　１００のダイレクト実行で

110～160：メニューの表示フォーマットです。全て英文になっていますが，カナ文字を使って適当な日本文に直してください。一応テレフォンリストとしています
１．ＷＲＩＴＥ；ファイルへの書き込み。
２．ＤＥＬＥＴＥ；データからの抹消。
３．ＲＥＡＤ　；ファイルの読み出し。
４．ＣＬＥＡＲ　；全てのデータを抹消する。

170：メニューの番号を選びます。０，５以上を選ぶと２００行からを実行するので，１９０行にＧＯ１００を入れると安全です。

180：選んだ番号により，それぞれの作業ルーチンにジャンプします。

```
200 REM *** WRITING ***  ……メニューで1を選んだとき，ここに来ます。
210 IF P>40 THEN GOTO 1600  ……配列の容量を超えると書き込みできません。
220 GOSUB 800  ……800行からは，名前を入力するサブルーチンです。
230 IF N$<=CHR$($F) THEN GOTO 1640  ……名前を入力せず，[RETURN] を押すと，メニューに戻れます。
240 GOSUB 850  ……850行はデータを入力するサブルーチン。
250 LET P=P+1  ……1人ファイルが増えたことを意味します。
260 GOSUB 300  ……300行のサブルーチンにジャンプ
270 GOSUB 900  ……900行のサブルーチンにジャンプ
280 IF R$="YES" THEN GOTO 200  ……書き込みを続けるときは YESを入力する。
290 GOTO 100  ……続けなければメニューに戻る。
300 REM ** SEARCH (K=1) **  ……220行で入力した名前を入れるファイルの場所をさがす。
310 FOR Z9=1 TO 1  ……ダミーループ。64頁参照。
320 FOR I=1 TO P  ……データの数（人数分）だけループをくり返す。
330 IF N1$(I)<N$ THEN GOTO 360  ……ファイルの名前が小さかったら，次のファイルを調べる。
340 IF N1$(I)=N$ THEN IF LEN(N1$(I))<=
LEN(N$) THEN GOTO 360  ……ファイルの名前が入力した名前と等しいときは，長さを調べ，ファイルが小さいか，等しいときは次を調べる。従って全く同じ名前のときは，新しいデーターほどファイルの後の方に入る。
350 GOTO 400
360 NEXT I
400 REM *** INSERT ***  ……名前が大きさの順になるようにファイルに挿入する作業。
410 FOR J=P TO I STEP -1  ……ファイルを書き込むときは，データの後の方からずらしていく。そうしないとデータが消えてしまう。
420 LET N1$(J+1)=N1$(J)
430 LET D1$(J+1)=D1$(J)
440 NEXT J
450 LET N1$(I)=N$  ……}
460 LET D1$(I)=D$  ……} 最後に入力したデータをファイルに書き込む。
470 NEXT Z9
480 RETURN

500 REM *** DELETE ***  ……メニューで2を選ぶとここに来ます。
510 CLEAR
520 GOSUB 800  ……名前を入力します。
530 IF N$<=CHR$($F) THEN GOTO 1640  ……名前を入力せず [RETURN] などのときは1640行に。
540 GOSUB 1000  ……抹消するファイルをさがすために1000行に
550 IF P1>0 THEN LET CURSOR=1,10:LET P
=P-1:PRINT N$;" WAS DELETED."  ……P1はデリートをしたとき1，ファイルに名前と同じデータがないときは0。1のときこの作業をする。プリント。
560 GOSUB 900  ……同じデリート作業を続けるかどうかを問い合わせるルーチン。
570 IF R$="YES" THEN GOTO 500  ……"YES"なら500行。
580 GOTO 100  ……でなければメニューに戻る。

600 REM *** READ ***  ……メニューで3を選ぶところに来る。
610 CLEAR
620 GOSUB 800  ……表示したい名前を入力する。
630 IF N$<=CHR$($F) THEN GOTO 1640
635 IF N$="* ALL FILE" THEN LET N$=CHR
$(4)  ……おまじない！ *を入力すると，全てのリストを表示する。
640 CLEAR
650 GOSUB 1000  ……表示する名前をファイルからさがすため1000行に。
660 GOSUB 900  ……他の名前のファイルを読むかを問い合わせるルーチン。
670 IF R$="YES" THEN GOTO 600  ……"YES"なら続ける。
680 GOTO 100  ……[RETURN] ならメニューに戻る。

700 REM *** CLEAR ***  ……メニューで4を選ぶとここに来る。
710 CLEAR :LET CURSOR=1,10
720 GOSUB 900  ……クリアーをするのかを入力するサブルーチンにジャンプ。
730 IF R$<>"YES" THEN GOTO 100  ……"YES"でなければメニューに戻る。
740 PRINT :PRINT "SURE ? IF CLEAR ALL
FILE THEN"  ……"YES"なら本当に全てをクリアーしていいのかもう一度念を押す。
750 PRINT :INPUT " HIT (CLEAR) AGAIN "
,R$  ……答えを要求します。
760 IF R$="CLEAR" THEN GOTO 20  ……もし本当に"CLEAR"なら，20行のCOLDスタートに戻る。
770 GOTO 100

800 REM *** IN NAME ***  ……名前を入力するサブルーチンです。文字列変数には32字まで代入できますが，READのときの画面表示のフォーマットを16字までを規準としています。一応16字までの名前を入力してください。
810 CLEAR
820 LET CURSOR=1,10:INPUT "NAME ",N$
830 RETURN
850 REM *** IN DATA ***  ……データを入力するサブルーチンです。例 電話番号，住所等，これも一応16字以内として入力してください。
860 PRINT :INPUT " DATA ",D$
870 RETURN
```

900 REM *** IN REPLY ***	作業が終ったとき，もう一度それらの作業を続けるかを問い合わせるためのサブルーチンです。
910 PRINT :PRINT "IF CONTINUE TO ";R1$ (K);", HIT (YES)　　　"	
920 PRINT "　　　　OTHERWISE HIT (CR) K EY. ":PRINT	CRとは RETURN キーのことで　もちろん，YES以外なら何でもいい。 RETURN だけでもOK。
930 INPUT R$	
940 RETURN	
1000 REM ** SEARCH (K=2,3) **	DELETE，READのときだけ，このルーチンで作業する。
1010 FOR Z9=1 TO 1	３１０行と同じ意味のダミーループです。
1020 LET P1=0:LET K1=K-1	初期条件，P１はデリート，読み出しを行なったときに１以上。
1030 FOR I=1 TO P	ファイルを１からP（人数）分だけめくるという感じです。
1040 IF N1$(I)<N$ THEN GOTO 1100	ファイルの名前が目的の名前より小さければ，次のページ。
1050 IF N1$(I)>N$ THEN GOTO 1110	ファイルの名前が目的の名前より大きければページをめくるのをやめる。
1060 ON K1 GOTO 1070,1090	
1070 IF LEN(N1$(I))=LEN(N$) THEN GOTO 1 200	ファイルの名前と目的の名前が等しくかつ文字列の長さも等しい条件のとき１２００行に無条件ジャンプ。
1080 GOTO 1100	等しかったけど,長さが違うので何もせず,1100行。
1090 IF LEN(N1$(I))>=LEN(N$) THEN GOSUB 1300	READのとき，名前が等しく，長さが目的とするファイルより長い名前を表示する。サブルーチンにジャンプ。これにより，イニシャル，姓などによる検索が可能です。
1100 NEXT I	
1110 IF P1=0 THEN CLEAR :LET CURSOR=1,1 0:PRINT N$;" IS NOT FILED."	P１＝０，すなわち，目的の名前がファイルにないとき，「N$はファイルにない」。と表示します。
1120 NEXT Z9	１０１０行と対になっているダミーループ。
1130 RETURN	
1200 REM *** DELETE ***	１０７０行の条件により，ここにジャンプして来ます。
1210 FOR J=I TO P	デリートのときは，消した人の名前のあったところから後のファイルを順につめてきます。
1220 LET N1$(J)=N1$(J+1)	
1230 LET D1$(J)=D1$(J+1)	
1240 NEXT J	
1250 LET P1=P1+1	P１＝１でもOK。ファイルを消したというサインです。
1260 GOTO 1110	
1300 REM *** PRINT OUT ***	READのとき，１０９０行の条件により，ここにサブルーチンジャンプしてきます。
1310 LET P1=P1+1:LET W1=LEN(N1$(I))	
1320 IF P1=1 THEN GOSUB 1500	P１＝１，すなわち，最初にプリントするときだけ。
1340 PRINT N1$(I); :PRINT TAB(16-W1);D1$ (I)	データを１６字までと仮定してこのようにしています。３２字までのデータを入力したいときは，LEFT$，RIGHT$などにより，プリント様式を工夫してください。
1350 FOR L=1 TO 32:PRINT CHR$($85); :NEX T L	
1360 RETURN	
1500 REM *** PRINT FORMAT ***	最初にプリントするとき，この作業ルーチンに来ます。
1510 PRINT "NAME　　　　　　DATA"	
1520 FOR L=1 TO 32:PRINT CHR$($FA); :NEX T L:PRINT	２本のケイをひく作業です。
1530 RETURN	
1600 REM ** RETURN TO MENU **	P>４０の条件のとき,つまり，配列宣言した分だけファイルに書き込まれていると，ここにきます。
1610 CLEAR :LET CURSOR=0,10	
1620 PRINT "FILE FULL! IMPOSSIBLE TO WR ITE."	
1630 GOTO 1660	
1640 CLEAR :LET CURSOR=0,10	WRITE，DELET，READで名前を入力せず，RETURN だけをキー入力したときここに来る。
1650 PRINT "SOON RETURN TO MENU WITH NO JOB!"	
1660 REM *** DELAY ***	２～３秒表示してメニューに戻るため時間待ちのルーチンです。
1670 FOR T=1 TO 1500:NEXT T:GOTO 100	
2000 END	
5000 REM *** FOR DEBUG *** 5010 FOR L=1 TO P+1:PRINT N1$(L); :PRINT TAB(16-LEN(N1$(L)));D1$(L):NEXT L:END	BREAK で中断したときなどGO５０００によりファイルを全て表示することができます。

●GOTOステートメント

プログラムの流れ，ステートメントの実行順序を変えるステートメントです。

```
GOTO　〈行番号〉
        └────跳び（ジャンプ）先の行番号

　TOを省略し，GOでもよい。
```

例57　無条件にプログラムの流れを変えます。

```
10　LET　A＝3
20　GOTO　50 ……………………… 50行めにジャンプ（跳ぶ。）
30　PRINT　A，A＊A ………… ┐
40　STOP ……………………………┘実行されない。
50　PRINT　A，A＊A＊A ……… AとAの3乗をプリント。
60　END
```

例58　CURSORステートメントを使った例です。U（UP），D（DOWN）R（RIGHT），L（LEFT）キーによる，画面中央から上下左右にダイアのマークで図を描くことができます。E（END）キーを押すと画面が消えて終りです。

```
10　CLEAR
20　X＝16：Y＝12 ……………………………… 初めの◆の位置
30　CURSOR＝X，Y
40　PR　CHR$（$97）； ……………………… ◆マークをプリント
60　Z$＝INKEY$：IF　Z$＜CHR$（1） …… キー入力，
　　GOTO　60
70　IF　Z$＝"U"　GOTO　130
80　IF　Z$＝"D"　GOTO　140
90　IF　Z$＝"L"　GOTO　150
100　IF　Z$＝"R"　GOTO　160
110　IF　Z$＝"E"　GOTO　200
120　GOTO　60
130　Y＝Y－1　：　GOTO　30
140　Y＝Y＋1　：　GOTO　30
150　X＝X－1　：　GOTO　30
160　X＝X＋1　：　GOTO　30
200　CLEAR ………………………………………… 全画面を消去，カーソルを
                                              左上すみに戻し終了します
500　END
```

●GOSUB……RETURNステートメント

GOSUBステートメントは，プログラムの実行をサブルーチンの先頭の行番号へ跳ばすステートメントです。

サブルーチンとは，一つ以上のステートメントから構成され，ある決められた作業を実行するプログラムです。サブルーチンの最終行には，RETURNステートメントをおき，GOSUBステートメントの次のステートメントにプログラムの流れを戻します。

GOSUB　〈行番号〉
　　　　↑────この行番号で示されるサブルーチンへ跳ぶ。

GOSと省略してもよい。

RETURN　　GOSUBで呼ばれた次の行番号にもどる

RETと省略してもよい。

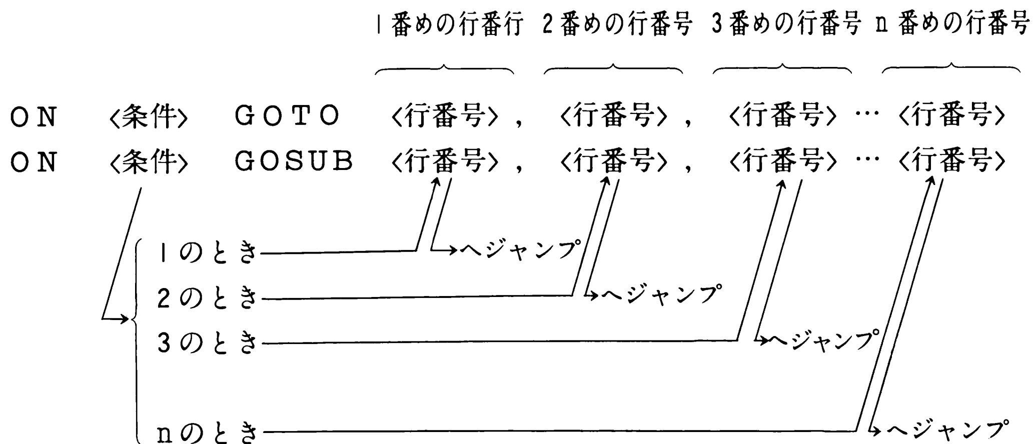
例59　約1秒間隔のカウント音と10秒毎のダブルカウント音を出します。

```
 10 REM 10ビョウカウンター
 20 LET A=0
 30 LET A=A+1
 40 GOSUB 200
 50 CLEAR ………………前の表示を消し同位置に表示する。
 60 PRINT A
 70 GOSUB 300………………カウント音のサブルーチン。
 80 IF A<10 GOTO 30
 90 GOSUB 300………………10秒毎のカウント音。
100 GOTO 20………………ループを初期状態に戻すために。
200 REM ** 1SEC ***……┐
210 T=TIME              │
220 IF TIME=T GOTO 220  ├単に1秒待つだけのサブルーチン。
230 RETURN……………………┘
300 REM ** ピ ッ ピ ッ **
310 MUSIC P0シ
320 RETURN
400 END        このプログラムを止めるときは BREAK/RESET キーを押す。
```

同一の作業を何回も繰返し行うようなときは，そのプログラムをサブルーチンにすることにより能率的にプログラムを作ることができます。

●ON…GOTO/ON…GOSUBステートメント

ＯＮとＧＯＴＯ（またはＧＯＳＵＢ）との間で成立する条件によってＧＯＴＯ（またはＧＯＳＵＢ）に続いて書かれるいずれかの行番号に分岐するステートメントです。



1．〈条件〉は〈式〉，〈数値変数〉，〈算術関数〉，〈文字取り扱い関数の中で＄のついていない関数〉

2．〈条件〉が小数のついた値のときは小数点以下を切り捨てた整数値。

3．〈条件〉が行番号の数以外のときは次の行番号を実行。

例　200　ON　A　GOTO　1000，1010，1020
　　210　PRINT　B＄

の場合，A＝0またはA≧4のときは行番号210を実行。
（A≦0のときA＝0となる）

4．ON　……　GOSUBのとき跳び先はサブルーチン形式であること。
すなわち，処理の終りにRETURNがあること。

例60

```
 10 CLEAR
 20 FOR A=1 TO 7
 30 ON A GOTO 50, 60, 70,   ……〈条件〉AはFOR-NEXTループ
    80, 90, 100                 1～7まで順次変化する。
 40 PRINT "END" :END……………A=7のとき：END
 50 PRINT "J";
 60 PRINT "U";
 70 PRINT "N";
 80 PRINT "I";
 90 PRINT "O";
100 PRINT "R";
110 PRINT
120 NEXT A
```

例61

```
10 INPUT "1 カラ 6 マデノ スウジ
     ヲ ドウゾ", A
20 CLEAR
40 FOR I=1 TO A
50 PRINT TAB (12);
60 ON I GOSUB 100, 110, 120,
   130, 140, 150
70 NEXT I
80 PRINT "***************
   ***************"
90 GOTO 10
100 PRINT "!!! 1 !!!" :RETURN
110 PRINT "*** 2 ***" :RETURN
120 PRINT "### 3 ###" :RETURN
130 PRINT "$$$ 4 $$$" :RETURN
140 PRINT "%%% 5 %%%" :RETURN
150 PRINT "&&& 6 &&&" :RETURN
```

例62 10進数を16進数に変換します。16進1桁の場合です。2桁だと……。

```
10 CLEAR
20 INPUT "0 カラ 15 マデノ スウジ"
   , D, "ハ",
30 IF D>15 THEN GOTO 10
40 IF D>=10 THEN GOSUB 100
50 IF D<10 THEN LET D$=RIGHT$--- D$=STR$(D)
   (STR$ (D), 1)                 だと。30ページ参照
60 PRINT "$"+D$
70 PRINT
80 GOTO 20
100 ON D-9 GOTO 110, 120, 130,-------- D-9が条件です。
    140, 150, 160
110 LET D$="A" :RETURN---------------┐
120 LET D$="B" :RETURN               │
130 LET D$="C" :RETURN               │ RETURNで,
140 LET D$="D" :RETURN               │ 50行に戻る。
150 LET D$="E" :RETURN               │
160 LET D$="F" :RETURN---------------┘
```

●FOR…TO…STEP/NEXTステートメント

FORとNEXTではさまれたステートメントをくり返して実行するステートメント。

FOR 〈変数〉 = 〈初期値〉 TO 〈最終値〉 STEP 〈キザミ幅〉※

〈変数〉を〈初期値〉から〈最終値〉まで〈キザミ幅〉のステップで変えて，FORとNEXT〈変数〉との間に書かれたステートメントをくり返して実行する。

NEXT 〈変数〉

1. 〈変数〉には配列（添字付変数）を設定できない。
2. 〈初期値〉，〈最終値〉，〈キザミ幅〉は式として書くことができる。
3. 〈キザミ幅〉は負であってもよい。
4. STEP〈キザミ幅〉を省略すると，STEP 1 とみなして実行する。
5. 〈変数〉はループ内では原則として変更できない。変更すると，変更後の値と〈最終値〉との間で〈キザミ幅〉のステップでループを作る。
6. FOR……NEXTのループの中からGOTOで抜け出ることができる。
7. FOR……NEXTのループの途中に，GOTO，GOSUBで跳び込むことはできない。
8. 多重ループを構成するときは，入れ子構造になっていなければならない。
   （**例68** **例70** 参照。）

※〈キザミ幅〉が正数のときには〈変数〉の値が〈最終値〉よりも大きくなると次の処理に，〈キザミ幅〉が負の数のときには〈変数〉の値が〈最終値〉よりも小さくなると次の処理に移ります。

例63

```
10 FOR A=1 TO 20 ……STEPを省略するとSTEPは1。
20 PRINT A,A^2
30 NEXT A ………………………Aが20になるまでループをくり返す。
50 END
```

例64 上記の例は，下記のように書き直すことができます。

```
10 LET A=1 ………………………〈初期値〉に相当
20 PRINT A,A^2
30 LET A=A+1 …………………〈キザミ幅〉に相当
40 IF A<=20 GOTO 20 ……Aが20以下なら20に戻る。
50 END
```

上の2つの例で，ダイレクト実行により PRINT A を行ってみてください。ループを抜け出るときの変数の値が重要になることがあります。

例 65　ＳＴＥＰ　によって，変数がどう変るかを確めてください。

```
10 FOR J=5 TO 10 STEP 2
20 PRINT J,
30 NEXT J
40 END
```

例 66　〈初期値〉，〈最終値〉，〈キザミ幅〉は式が使えます。

```
10 P=1:Q=100:R=3:S=5
20 FOR I=P TO Q*S/2 STEP R+S
30 PRINT I,
40 NEXT I
50 END
```

例 67　〈初期値〉が〈最終値〉よりも大きく，〈キザミ幅〉が正の場合でも１ループだけは実行します。

```
10 CLEAR
20 FOR B=10 TO 1 STEP 5 ･･･┐
30 PRINT B,B*B              │ 不条理なループ
40 NEXT B ･･････････････････┘
50 PRINT B
99 END
```

これは，〈初期値〉１０による１回のループは不条理ではなく、ＮＥＸＴ　Ｂによって始めて　Ｂが〈初期値〉と〈最終値〉との間にないことが判断されるからです。

例　小さい〈キザミ幅〉でＦＯＲ　－　ＮＥＸＴ　のループをくり返すと，最終桁に誤差を生じることがあります。

```
10 CLEAR･･････････････････････RUNで10行から実行。
20 FOR A=1 TO .999 STEP･･････0.XXの0は省略できる。
   -.00005
30 PRINT A,%7;A
40 NEXT A
50 END
60 CLEAR･･････････････････････RUN60でここから実行
70 FOR A=.999 TO 1 STEP
   .00005
80 PRINT %11;A;%7;A･･････････30行との違いに注意
90 NEXT A
100 END
```

例68　入れ子構造とはループが交叉していないことを言います。

```
10 FOR Y=0 TO 9
20 A=1
30 A=A+5*Y
40 B=A+4
50 FOR X=A TO B
60 PR %5; X,
70 NEXT X
80 PR:PR
90 NEXT Y
100 END
```

交叉していない

60行：コンマを必ずつけること。

80行：改行と，一行あけるためです。

よく考えてください……。

このプログラムが理解できたら，横（X）を7段にしたプログラムを作ってください。

例69　11時間59分59秒までカウントするタイマーです。TIME関数は水晶振動子の周波数を分周しているため，正確な時間計測ができます。

```
10 CLEAR
20 INPUT "START", A$
30 FOR H=0 TO 11
40 FOR M=0 TO 59
50 FOR S=0 TO 59
60 CLEAR :LET CURSOR=10, 10
70 PRINT H; CHR$ ($E0) ; M;
   CHR$ ($E1) ; S; CHR$ ($E2)
80 LET T=TIME
90 IF TIME=T THEN GOTO 90
100 NEXT S
110 NEXT M
120 NEXT H
130 END
```

60行：プリント位置を全てCURSOR文で決めれば，CLRは不要。

ループ1（90→90），ループ2（50〜100），ループ3（40〜110），ループ4（30〜120）

例70　ループの途中にGOTOで跳び込むことはできません。

```
10 FOR A=0 TO 100
20 IF A*A>150 GOTO 50
30 NEXT A
40 FOR B=0 TO $FF STEP $F
50 PRINT A, B
60 NEXT B
99 END
```

×これはダメ。

例71　ＦＯＲ－ＮＥＸＴのループから抜け出ることは出来ますが，注意が必要です。この例ではループがＮＥＸＴ　Ｉに出ているのでＯＫですが。

```
10 PRINT "START"
20 FOR I=1 TO 16
30 FOR J=1 TO 5
40 PRINT I, J
50 IF I=J GOTO 70
60 NEXT J
70 NEXT I
80 PRINT "END"
99 END
```

ループからＧＯＴＯで抜け出る。

例72　この例ではループからの抜け出しが重なって多重ループを作ったことになり，エラーを生じます。実際にＲＵＮしてためしてください。

```
10 CLEAR
20 LET A=0
30 GOSUB 100        ←⑤
40 GOTO 30
100 LET A=A+1       ←①
110 FOR I=1 TO 20   ←④
120 IF I=A GO 140   ←②
130 NEXT I
140 PRINT A         ←③
150 RETURN
```

ループから抜け出ることはできるが，これが重なると多重ループとなる。

このようなエラーを避けるためには，つぎのように，ダミーループを入れることにより多重ループになるのを回避できます。上の例に追加します。

```
105 FOR D=1 TO 1    ⑥
145 NEXT D
```

１０５行から１４５行までを１回だけ通過するループです。

このループ ⑥ の中にループ ②, ③ が入っているため，ループ ③ がたかだか3重ループ止まりですみます。ベーシックマスタージュニアは１５重までの多重ループを構成することができます。

●READ/DATA/RESTOREステートメント

ＲＥＡＤは，プログラムの中に設定してあるデータ（数値または文字列）を変数に代入するステートメントで，ＤＡＴＡはＲＥＡＤステートメントで読まれるデータを設定するステートメントです。また，ＲＥＳＴＯＲＥは，ＲＥＡＤで読まれたデータを再び読み込むときに使うステートメントです。

ＲＥＡＤ　〈変数〉，〈変数〉，〈変数〉，………，〈変数〉

ＤＡＴＡ　〈数値または〝文字列〟〉，〈数値または〝文字列〟〉………

↓

文字列は〝　〟でくくる。

例）ＤＡＴＡ　〝イロハ〟，〝ＢＡＳＩＣ　ＭＡＳＴＥＲ〟

1．ＤＡＴＡ文で設定されたデータの数は，ＲＥＡＤ文で代入される変数の数，（ＲＥＡＤ文がループの中でくり返されるときは，そのくり返しの数）と等しいかそれ以上でなければならない。

2．ＤＡＴＡ文はプログラムのどこにあってもよく，また分散して設定してもよい。

3．〈変数〉への〈数値または文字列〉の代入は，ＲＥＡＤ文の実行される順番に従い，プログラムの中でＤＡＴＡ文で設定される若い方から順次代入されていく。

4．〈変数〉への〈数値または文字列〉の代入は，必ず〈変数〉タイプとデータのタイプが一致しなければならない。

例）ＲＥＡＤ　〈数値変数〉，〈文字変数〉

ＤＡＴＡ　〈数値〉，〈文字列〉

ＲＥＳＴＯＲＥ

ＲＥＳＴＯＲＥの次にくるＲＥＡＤ文は，プログラムの最初のＤＡＴＡ文からデータを変数に代入する。

例73

```
10 READ A:PRINT A
20 READ B:PRINT B
30 READ C:PRINT C
40 DATA 123,456,789
50 END
```

例74　数値データと数値変数，文字データと文字変数の順番が対応していること。

```
10 READ A,A$,B,B$
20 DATA 12,"BASIC",34,"ジュニア"
30 PRINT A;A$;B;B$
40 END
```

例75 何はともあれ，キー入力して実行してみてください。

```
10  CLEAR
20  LET CURSOR=10,10
30  READ A
40  IF A=0 THEN END ……………… データに0があったら実行終了
50  PRINT CHR$(A);
60  GOTO 30
70  DATA 156,32,186,222,184,219
80  DATA 179,187,207,32,156,13,13
90  DATA 7,181,188,207,178,13,0
```

例76 RESTOREは同じデータを何回も読みたいときに必ず必要です。

```
10  FOR I=1 TO 10
20  RESTORE ……………………………… 65行，75行の位置でもOKです。
30  FOR J=1 TO 10
40  READ A$
50  PRINT A$;
60  NEXT J
70  PRINT
80  NEXT I
90  DATA"2","0","␣","R","E","S","T","O","R","E"
100 END
```

例77 10進または，16進数を8ビットの2進数表示に変換するプログラムです。

```
10  CLEAR
20  INPUT N ……………………………… 10進または16進4桁までOK。
30  PRINT TAB(8);CHR$($B) … プリント位置をNにつづけて表示するため。
40  RESTORE ……………………………… これがないと一回きり。
50  READ D ………………………………… 120行のDATAを読む。
60  IF(D=0)*(N>0)THEN PRINT …… データ0でかつ余りがあれば
    "␣␣␣OVER"
70  IF D=0 THEN PRINT:PRINT …… データ0なら1回の実行完了。
    :GOTO 20
80  LET N=N-D
90  IF N>=0 THEN PRINT 1; …  }
100 IF N<0 THEN LET N=N+D …… } よく考えてください。
    :PRINT0;
110 GOTO 50
120 DATA 128,64,32,16,8,4,2,1,0 …… 2^7, 2^6, …… 2^0, 0
```

●INPUTステートメント

キーボードから数値または文字を入力するステートメントです。

```
INPUT  〝文字列〟, 〈変数〉
                                    }  〝文字列〟はなくてもよい。
  IN  〝文字列〟, 〈変数〉と省略できる。
```

1．〝文字列〟，〈変数〉は，で区切って続けて書くことができる。

例）INPUT 〝A，A$〟，A，A$，〝H，H$〟，H，H$

2．INPUT文を実行し，入力を要求するときは ？ が出力され _ （カーソル）が点めつする。

3．入力要求と異なったタイプの入力を行うと〝ピー〟という警告とともに，？がもう一つ出力され，改めて入力要求の _ が点めつする。

例）INPUT A と数値の入力を要求しているのにアルファベットをキー入力したとき。

4．〝文字列〟があるときは文字列を表示した後？が出力され_が点めつする。

5．入力する文字列の先頭に空白（スペース）がある場合には，〝 〟（引用符）で囲ってください。

例78 入力要求のとき，？を出力しカーソルが点めつします。

```
10 INPUT A ……………… 数値の入力を要求しています。入力 [RETURN]
20 PRINT SQR (A)
30 END
```

例79 区切り記号（，）はPRINT文と意味が異なります。ご注意ください。

```
10 DIM A(4)
20 INPUT A(1), "+", A(2) ……入力 [RETURN] 入力 [RETURN]
30 LET A(3) =A(2) +A(1)
40 INPUT "コタエハ_", A(4) ……スペースは" "の中に入れる。
50 IF A(3)=A(4) GOTO 80
60 PRINT "チガ イマス"
70 GOTO 40
80 PRINT "ソウデ ス"
99 END
```

要求以外のキー操作に対しては？？が表示されますが，そのままつづけて正しい値をキー入力してかまいません。

●CLEARステートメント

表示画面を消去し，カーソルを画面左上端に設定するステートメントです。

CLEAR　　　　　　CLR　と省略できる。

1．表示画面を消去し，カーソルを画面左上端に設定する。
2．プログラムは消去されない。　⟷NEWコマンドはプログラムも消去される。

例80　TIMEは自動的に1秒ごとに増えつづけています。

```
 10 CLEAR·························最初に画面をクリアーして。
 20 FOR I=10 TO 1 STEP -1
 30 PRINT "アト";I;CHR$($E2);
    "デ ガ メン キエマス"
 40 LET T=TIME··················Tに時間を代入
 50 IF TIME=T THEN GOTO 50····TIME(時間)が40行のTと同
 60 NEXT I                       じなら分岐。
 70 CLEAR·························ここで消える!
 80 LET TIME=0··················TIMEを0にする。
 90 IF TIME<10 THEN GOTO 90···10秒たっていなければ分岐。
100 LET CURSOR=10,16
110 PRINT "END"
120 END
```

例81　ダイレクト実行でCLEARを実行すると大変便利です。特に図形などの表示をしたあとでLISTを見たいとき，カーソルが画面中央に残ったままだとカーソル位置からLISTが表示されるため，画面の表示と，リストがごちゃごちゃになることがあります。こんな状態で画面エディターは使えません。そのようなとき，

CLR：L RETURN　　マルチステートメントのダイレクト実行を行ないます。
CLR RETURN　L RETURN　としてももちろんOK。その他
CLR：R RETURN　　などもOK。CLEARは，プログラムの中でも使えますが，コマンドと考えてどしどし使ってください。

ヒント　省略形をどんどん使おう。

LETなどいちいち入力するのはアホくさい。CURSORなんてミスキーしそうなスペルのステートメントは ! で十分。PRINTは?で一発。第一，ミスキー入力したときのあの，ピーッという警告音と，SYNTAX ERRORという表示にはうんざり。
どんどん省略形を使おうではないか。ついでに，省略形で入力してもソースプログラムは短かくならない。一行入力して RETURN キーを押すごとに中間言語にほんやくしているからだ。このため一行入力ごとに例のSYNTAX ERRORのお世話になることができるとも言える。

●DEF FNxステートメント（ユーザー定義関数）

ユーザー自身が式を組み合わせ，１つの関数としてプログラム中で定義するステートメントです。

DEF　FNx＝〈式〉　（引数なし）
DEF　FNx（仮引数）＝〈式〉

xはA～Z，A0～Z9までのうちの一つを表わす。

仮引数：右辺の〈式〉の中で用いられる引数を示す単純数値変数。関数として用いられるときは，実際の引数におきかえられる。

1．一度，DEF　FNxで定義すれば，あとは，FNxまたはFNx（X）の形で，〈式〉やPRINT文，LET文などで使える。
2．FNx（X）を使うステートメントの前で必ずDEFで定義しなければならない。
3．FNx（X）を二重定義すると，実行の順序に従い後の方で定義した関数が有効となる。

例82　ベーシックマスタージュニアの三角関数はラジアンで計算されるため，角度で計算するには不便です。まず，ラジアンを角度に換算する式を定義します。

```
10 DEF FNR=PAI/180 ……………………2πラジアン=360°
20 INPUT R
30 LET S=SIN(R*FNR)
40 PRINT S
50 END
```

例83　**例82** の30行めも一諸に定義すればもっと簡単です。

```
10 DEF FNS(R)=SIN(R*PAI/180)
20 INPUT R
30 PRINT FNS(R)
40 END
```

つぎに分，秒もそのまま入力できるユーザー関数を考えてください。

例84

```
10 CLEAR
20 DEF FNA(R)=4/3*PAI*R^3 ……………………4/3πr³
30 DEF FNB(R)=4*PAI*R^2 ……………………4πr²
40 INPUT "ハンケイ ニュウリョク",R
50 PRINT "キュウ ノ タイセキ    ";FNA(R)
60 PRINT "キュウ ノ ヒョウメンセキ";FNB(R)
70 END
```

●RND(X)関数／RANDOMIZEステートメント

ＲＮＤ（Ｘ）は疑似的な乱数を発生させる組み込み関数で，ＲＡＮＤＯＭＩＺＥは，ＲＵＮごとにＲＮＤ（Ｘ）による乱数発生を不規則にするステートメントです。

ＲＮＤ（Ｘ）	$0<X\leqq 1.70141183\times 10^{38}$
ＲＡＮＤＯＭＩＺＥ	ＲＮＤＭと省略できる。

1．ＲＮＤ（Ｘ）は０～Ｘ未満までの疑似的な乱数を発生する。
2．ＲＮＤ（Ｘ）は１回のＲＵＮの中では実行されるたびに不規則ではあるが，次のＲＵＮでは再び同じパターンの乱数が発生する。
3．ＲＡＮＤＯＭＩＺＥを使うとＲＵＮごとに不規則になる。
4．Ｘ≦０のときに，ＲＮＤ（Ｘ）＝０となる。

例85　ＲＵＮ　１０　または，ＲＵＮ　３０により，疑似的な乱数の発生の様子を確めてください。

```
10 PRINT RND(10)
20 END
30 RANDOMIZE
40 PRINT RND(10)
50 END
```

例86　１または２の乱数を発生させ，１と２とが何個ずつ発生するかを調べます。ついでに，作業にかかる時間を計測しました。

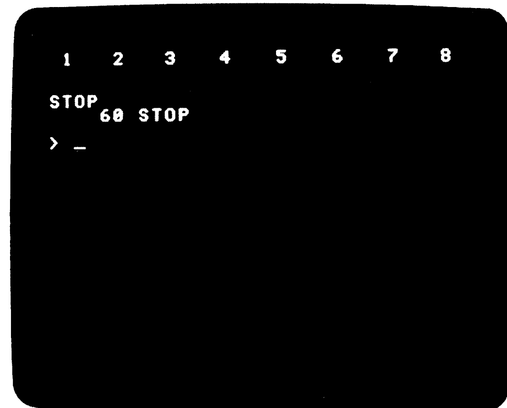
```
 10 TIME=0
 20 R=0:R1=0:R2=0
 30 R3=INT(RND(2))+1 ………INT(RND(2))は 0または1。
 40 ON R3 GO 50,60
 50 R1=R1+1:GO 100
 60 R2=R2+1:GO 100
100 R=R+1
110 IF R=1000 GO 150 ………ここで1000回までをカウント。
120 GO 30
150 PRINT R1,R2,TIME
160 END
```

●STOPステートメント／(CONTINUEコマンド)

プログラムの途中に，行番号とこのキーワードを書き，プログラムの実行を中断するステートメント。
STOPステートメントにより一時停止したプログラムを実行させるときは，コマンドCONTINUEを使います。(CONTINUEはCONTまたはCと省略できる。)

例88

```
 5 CLR
10 LET I=0
20 LET I=I+1
30 PRINT %3; I,
40 IF I< 8 GOTO 20
50 PRINT
60 STOP
70 GOTO 10
99 END
```



●ENDステートメント

メインプログラムの終りに書き，プログラムの実行を終らせる。

例89

```
10 FOR N=1 TO 1000
20 NEXT N
30 PRINT "END"
40 END
```

ヒント　プログラムのデバッグのやり方

1．プログラムの途中にデバッグ用のステートメントを入れる。このとき，デバッグ用のステートメントであることを他のステートメントと区別できるように行番号の下1桁を他にない数，たとえば9にしておくと便利です。

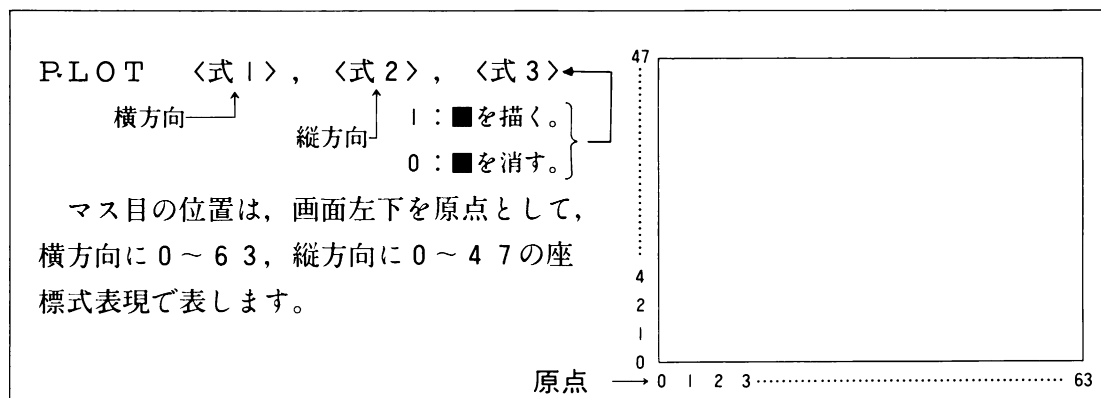
```
 10 INPUT A
 20 B=2*A+34+3*(A-4)
 29 PRINT "B=" ;B
 30  :
999 END
```

└デバッグ用ステートメント
完成したらこの行削除。

2．左記の例で，29 STOP としてプログラムを実行させます。STOPしたら，ダイレクト実行で PRINT B とすると同じようにこのときのBの値が表示されます。続いてCONTとすればプログラムの実行が再びスタートします。

3.7. 特殊なステートメント／●PLOTステートメント

ＰＬＯＴは，横６４，縦４８のマス目に分解した画面のマス目を，■マークでぬりつぶしたり，ぬりつぶしてあるものを消したりしてモザイク状の図形を描くステートメントです。



1．座標指定〈式1〉，〈式2〉が表示範囲をはみ出す場合

a．〈式1〉が６３よりも大きいとき，

〈式1〉＝〈式1〉－ＩＮＴ（〈式1〉／６４）×６４の計算を行ない，画面の範囲内にプロットできるように変換して出力する。

b．〈式2〉が４７よりも大きいとき，

〈式2〉＝〈式2〉－ＩＮＴ（〈式2〉／４８）×４８の計算を行ない，画面の範囲内にプロットできるように変換して出力する。

2．〈式3〉の範囲

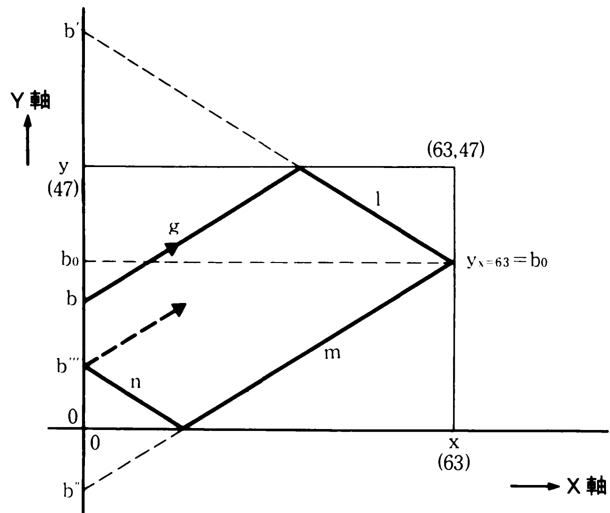
〈式3〉＝０　以外のとき…■をプロットする。（通常は〈式3〉＝１とする。）

〈式3〉＝０　のとき………上記でプロットした■を消す。

例90　２０行の２次関数のグラフを描くプログラムです。Xの２乗の係数は，グラフを画面いっぱいに表示するための工夫です。

```
10 FOR X=0 TO 63
20 LET Y=X*X/4*3/63
30 PLOT X,Y,1
40 NEXT X
90 END
```

例91 図形がカベに当って反射するような2次元図形移動のプログラムを考えてみます。



直線の方程式は一般的に，

$y = ax + b$　　a ：勾配，傾き
　　　　　　　　b ：Y軸との交点

直線gが，反射してl，m，nの直線になったとすると，Y軸との交点，b′，b″，b‴は次のようになります。

①…$b' = b + 2(y - b) = 2y - b$
②…$b'' = b' - 2(b' - b_0) = 2b_0 - b'$
③…$b''' = -b''$

aは反射するたびに＋－が反転するだけです。

以上のことが理解できたら次のプログラムをよく見てください。40行，60行，70行を注目してください。プログラムは全て省略形で書きます。

```
  5  Y=0
 10  IN A, C
 20  PR "SLOPE=";A;"/"; C
 30  IN B:CLR ………………………… 画面消去
 40  FOR D=1 TO -1 STEP -2 …D=1又はD=-1となる。
 50  A=A*D
 60  B=B*D+2*Y*(D<0)                D=1 ……  (D<0)=0
                                             (D>0)=1
 70  FOR X=63*(D<0) TO
     63*(D>0) STEP D                D=-1 …  (D<0)=1
                                             (D>0)=0
 80  Y=A*X/C+B
 90  IF Y<=0 Y=0:A=-A:B=-B
100  IF Y>=47 Y=47:A=-A:B=2*Y-B
110  PLOT X, Y, 1
150  NEXT X                     y=ax+bのbは
160  NEXT D                       ①式が100行
170  A=-A:GO 40                   ②式が60行
200  END                          ③式が90行 になっています。
```

勾配aは，A／Cで，Y軸との交点はBで入力します。例えば，A=1，C=1，B=1などとして実行してみてください。つぎに図形を移動させるには，

```
140  PLOT X, Y, 0
```

でOKです。移動が早く，チカチカしすぎるときは，

```
120  FOR F=1 TO 10
130  NEXT F
```

を追加してください。

このプログラムはCURSOR文にも応用できます。色々な応用を考えてください。

●POKEステートメント

メモリーの絶対番地に，データを書き込むステートメントです。

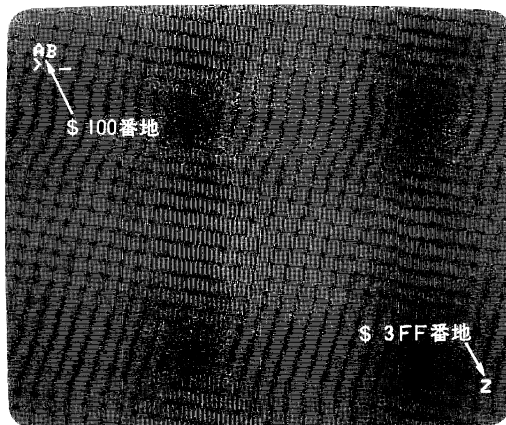
POKE 〈式1〉，〈式2〉，〈式3〉	〈式1〉に示される番地に〈式2〉で示される値の下位1バイトの値を書き込み，(〈式1〉+1)番地に〈式3〉で示される値の下位1バイトの値を書き込みます。

〈式3〉以降は1行79字となるまで〈式4〉，〈式5〉……と書き込み可能です。

例 92　表示画面に対応する絶対番地に直接データを書き込みます。

```
              A           ZのJISコード
              ↓           ↓
10   A=$41:B=$5A
15   CLEAR              BのJISコード
                             ↓
20   POKE  $100, A, A+1
30   POKE  $3FF, B
40   END
```

$100，$3FFは表示画面の左上，右下に対応するメモリーの絶対番地です。



例 93　POKEステートメントの特殊な使い方を考えてみました。47ページの図形を全て表示するプログラムです。1030行がポイントです。

```
1000  CLR:PR "0␣1␣2␣3␣4␣5␣6␣7␣ …スペースを入れる。
      8␣9␣A␣B␣C␣D␣E␣F"
1010  FOR  A=0  TO  $F ……………………$Fです。
1020  FOR  B=0  TO  $F
1030  X=B*$10+A ……………………座標を変換するためです。
1040  Y=A*$10+B:P=2*Y+$140
1050  POKE  P, X ……………………POKE文には変数が使える。
1060  NEXT  B
1070  NEXT  A
1080  CURSOR=0, 20 ……………………このステートメントが必要。
1100  END
```

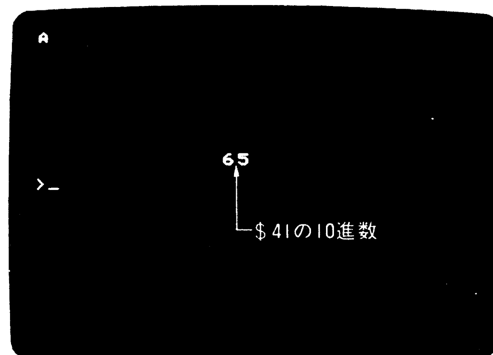
●PEEK(X)関数

メモリーの絶対番地データを読みとる関数です。

PEEK（〈式〉）	〈式〉で示された絶対番地の内容を読む。

例94　１バイトの内容を読むプログラム例です。

```
 5 CLEAR
10 POKE $100, $41
20 CURSOR=10, 10
30 PRINT PEEK ($100)
40 END
```



10行で＄１００番地に＄４１を書き，30行でその＄１００番地を読みます。

ヒント　10進数と16進数

ベーシックマスタージュニアは８ｂｉｔ（ビット）のパーソナルコンピュータです。８ｂｉｔとは２進数（バイナリーコード）で８桁であるということを表します。ジュニアの中では全てこの２進数によって処理されていますが，２進数では人間の感覚にピッタリこないのでこれを１６進数で表現しているわけです。例のドル記号（＄）で表わしているものです。例えば２進数で１１０１０１０１は１６進数では＄Ｄ５，１０進数では２１３ですが，次のような仕組みになっています。

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
２進数	１	１	０	１	０	１	０	１

上位４ｂｉｔ　　下位４ｂｉｔ

⇩　　⇩

１０進数………１３（×１６＋）５（＝２１３）

１６進表現……＄Ｄ　　＄５

＄Ｄ５

８ｂｉｔを上位４ｂｉｔと，下位４ｂｉｔとに分け，それぞれの４ｂｉｔを１６進数にしたものを１６進数的に表現しているわけです。

●CALLステートメント

ＢＡＳＩＣ言語で書いたプログラムの中から，機械語で書かれたプログラムを呼び出すためのステートメントです。機械語のプログラムはサブルーチン形式でプログラムの末尾がＲＴＳ（機械語39）命令（ＢＡＳＩＣではＲＥＴＵＲＮ〈ＲＥＴ〉に相当）になっていることが必要です。

> ＣＡＬＬ　〈機械語プログラムの先頭番地〉
>
> 機械語プログラムの処理をしたあと，機械語プログラムのＲＴＳ命令により，ベーシックプログラムのＣＡＬＬステートメントの次の行番号のステートメントに実行が移る。

機械語のプログラムでスタックポインタ（ＳＰ）を使う場合，使用に先だちスタックポインタを退避させ，ＲＴＳ命令の前で元にもどすプログラムを機械語プログラムの中に書いておく必要があります。

例95　モニターのサブルーチンを使って，実用的なベーシックのサブルーチンを作ってみます。

```
5000 REM *** ガ メン テンソウ ***
5010 POKE $3B, $01, $00
5020 POKE $3D, $03, $FF   詳しくはモニターのサブルー
5030 POKE $3F, $1A, $00   チンの項をご参照ください。
5040 CALL $F009
5050 CLEAR
5060 RETURN

6000 REM *** ガ メン ヨ ビ ダ シ ***
6010 POKE $3B, $1A, $00
6020 POKE $3D, $1C, $FF
6030 POKE $3F, $01, $00
6040 CALL $F009
6050 RETURN
```

このサブルーチンは，画面の表示エリアに相当するメモリーのエリア（＄１００～＄３ＦＦ）の７６８バイトの内容を，他のメモリーエリア(＄１Ａ００～＄１ＣＦＦ)にブロック転送し，必要なときに転送した画面を呼び出すというものです。

これらのサブーチンを交互に呼び出して点めつさせたり，しばしば使うグラフのケイなどをあらかじめ作っておき，グラフをプロットする前に呼び出すなどの応用が考えられます。（ガメンヨビダシは何度でも呼び出せます。）

●MUSICステートメント

音楽の音符を記号で入力し，これを実行すると記号入力した曲を自動演奏するステートメントです。

MUSIC〈音楽記号〉	MU〈音楽記号〉	
MUSIC〈文字列〉	MU〈文字列〉	と省略してもよい。
MUSIC〈文字変数〉	MU〈文字変数〉	

まず，音楽記号について説明します。

(1)音階　（記号：トレミフソラシ　＃　B）

音階	ド	レ	ミ	ファ	ソ	ラ	シ
キー入力する音階記号	ト	レ	ミ	フ	ソ	ラ	シ

半音	シャープ ＃	フラット b	＃，Bで指定した1音のみ有効となる。
キー入力する半音記号	＃	B	

例 96　後面の音量ツマミを音量が大となるようにして，ダイレクト実行で確めてください。

MUSIC [カナ] トレミフソラシ [RETURN]

MUSIC [カナ] トレミフソ [英数] B [カナ] ララ [RETURN]

←Bで指定した後の1音のみフラットが有効となる

(2)音程（記号：U，D）

音程は3オクターブまで指定できます。

音程記号　|←　Dで指定　→|←　無指定　→|←　Uで指定　→|

DトDレDミDフDソDラDシ　ト　レ　ミ　フ　ソ　ラ　シ　UトUレUミUフUソUラUシ

D，Uの指定は，すぐ後の1音に対してだけ有効です。

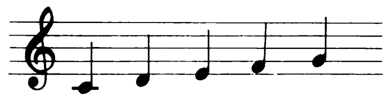
例 97　[英数] [カナ] キーは省略します。忘れないでキー入力してください。

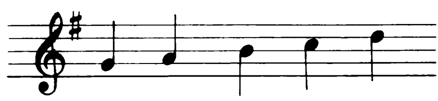
MUSIC　DソDラDシトレミフソラシUトUレUミUフUソ [RETURN]

(3) 移調（記号：M）

移調記号Mのあとに，移調したい調の主音のハ調読みを入れる。移調範囲はBト～#シ。電源ON後指定しないときは，ハ調。

移調記号は一たん指定すると，次に指定されるまでそのままの調を続ける。

例 98

MUSIC　トレミフソ RETURN

↑
無指定のときはハ長調。

MUSIC　Mソトレミフソ RETURN

↑
ト長調。

(4) 音の長さ　（記号：P0～P9）

指定した長さの音を出力します。

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
𝅘𝅥𝅰	𝅘𝅥𝅯	𝅘𝅥𝅯.	𝅘𝅥𝅮	𝅘𝅥𝅮.	𝅘𝅥	𝅘𝅥.	𝅗𝅥	𝅗𝅥.	𝅝
32分音符	16分音符	付点16分音符	8分音符	付点8分音符	4分音符	付点4分音符	2分音符	付点2分音符	全音符

電源ON後，特に指定しない場合はP5となる。

例 99

MUSIC　P5トレP3ミミP5フ#ソフP7ミ RETURN

(5) 休止符　（記号：P0R～P9R）

指定した休止符の長さだけ音を休止します。

P0R	P1R	P2R	P3R	P4R	P5R	P6R	P7R	P8R	P9R
𝅀	𝄿	𝄿𝅀	𝄾	𝄾𝄿	𝄽	𝄽𝄾	𝄼	𝄼𝄽	𝄻
32分休止符	16分休止符	付点16分休止符	8分休止符	付点8分休止符	4分休止符	付点4分休止符	2分休止符	付点2分休止符	全休止符

電源ON後，特に指定しない場合はP5となり，音階の後にRをつけるとP5Rとなる。

Ｐ０～Ｐ９を省略するとＲを指定する前で指定した音符の長さと長じ長さの休止符となる。

例 100

ＭＵＳＩＣ　Ｐ６ソＰ３ラＰ５シＲソＰ６ソＰ３フＰ５Ｒソシ RETURN

この前に指定されたＰ５の長さの休符（４分休符）　　４分休符　　４分音符

(6) 音の大きさ　（記号：Ｖ１～Ｖ５）

指定した大きさの音を出力します。最強音Ｖ５を１として，Ｖ１になるに従い$\frac{1}{2}$ずつ小さくなります。

音楽記号	Ｖ１	Ｖ２	Ｖ３	Ｖ４	Ｖ５
Ｖ５を１とした音の大きさの比	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	１

電源ＯＮ後，特に指定しない場合はＶ４。

例

ＭＵＳＩＣ　Ｖ５Ｐ６ソＰ３ラＰ５シＲＶ２ソＰ６ソＰ３フＰ５Ｒソシ RETURN

(7) 音色　（記号：Ｑ０～Ｑ４）

下図のような波形を出力し，音色に変化をつけることができます。

Ｑ０	Ｑ１	Ｑ２	Ｑ３	Ｑ4

音色は言葉では言い表しにくいので自分の耳で確めてみてください。電源ＯＮ後，特に指定しない場合はＱ０。

(8) 速さ　（記号：Ｔ１～Ｔ７）

７段階の速さ（テンポ）を設定できます。

Ｔ１	Ｔ２	Ｔ３	Ｔ４	Ｔ５	Ｔ６	Ｔ７
♩＝２００	♩＝１３３	♩＝１００	♩＝８０	♩＝６７	♩＝５７	♩＝５０

電源ＯＮ後，特に指定しない場合はＴ４。

例101 「喜びの歌」 (ベートーベン作曲)のプログラム。

```
10 PRINT TAB(5); "ヨロコビ ノウタ"
20 MUSIC MソT2V3ミミフソソフミレトトレミP6ミ
   P3レP5レR
30 MUSIC ミミフソソフミレトトレミP6レP3トP5ト
   R
40 MUSIC V4レレミトレP3ミフP5ミトレP3ミフP
   5ミレトレDソV5P7ミP5ミフソソフミレトトレミP6
   レP3トP5トR
99 END
```

注意

1. #, B, U, D·········指定したあとの1音に対してのみ有効となる。
2. 上記以外の音楽記号·········次の新しい指定があるまで連続して有効。

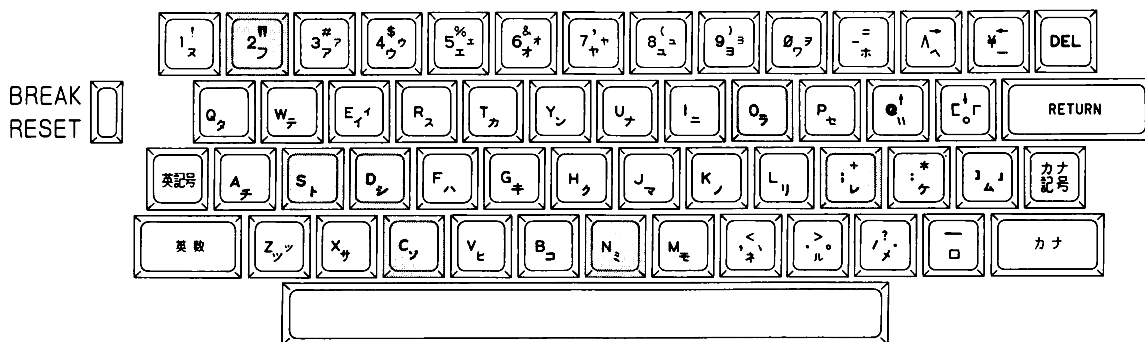
例 MUSIC レトレP5シBレミフレ#ミV5トレP3ミフ···········

Bは次の1音(レ)に対してのみ有効

#は次の1音(シ)に対してのみ有効

P5は次の新しい指定，P3がくるまで連続して有効。

ド→シのキー位置



カナ キーによるカナ文字と 英数 キーによる英数字との切り替えを忘れないように。

音楽記号のまとめ

1．音　　階	ド→ト，レ→レ，ミ→ミ，ファ→フ，ソ→ソ，ラ→ラ，シ→シで入力
2．基本使用法	MUSIC␣とキー入力した後，上記音程をカナにて入力　[例] MUSIC␣トレミフソラシ [RETURN]
3．半　　音	#(シャープ)，B(フラット)を音の前に付ける　[例] MUSIC␣ト #ト Bト…………
4．休　　符	Rと入れるとそれ以前に指定された長さだけ音を休止　[例] MUSIC␣トレミRRフソ…………
5．音　　程	D指定 (例DトDレ…)　無指定 (例トレミ…)　U指定 (例UトUレ…) 左のようにDとUで下と上を指定 例 MUSIC␣トレミUトミソラDシト……
6．移　　調	Mのあとに移調したい調の主音のハ調読みを入れることにより移調 (移調範囲　Bト～#シ) [例] MUSIC␣トレミフソ(指定しないときはハ調)　MUSIC␣Mソトレミフト(ト調)
7．音の長さ	Pと数字で32分音符から全音符まで10段階指定（P0～P9） P0　P1　P2　P3　P4　P5　P6　P7　P8　P9
8．音の大きさ	Vと数字で5段階指定　V1(最弱)～V5(最強)
9．音　　色	Qと数字で5種類指定　Q0(連続音)　Q1　Q2　Q3　Q4
10．速　　さ	Tと数字で7段階指定　T1：♩=200　T2：♩=133　T3：♩=100　T4：♩=80 T5：♩=67　T6：♩=57　T7：♩=50
11．そ の 他	(1) #，B，U，Dの指定はすぐその後の音にだけ有効，他の指定は次の指定が入るまで連続有効 (2) 上の6～10の指定を特に行わず入力すると下記が自動的に指定される 調=ハ調，音の長さ=P5(♩)，音の大きさ=V4(最強の1つ下)，音色=Q0(⊓連続音)，速さ=T4(♩=80)

ヒント　画面エディターを使ってMUSICを作ろう。

1．まず，MUSICとキーインした後，[カナ]モードにして音階をトレミフソでキー入力して[RETURN]を押します。

2．つづいて，画面エディターを使い，[英数]モードでP，V，Q，T，#，B，R，数字等を挿入します。このようにするといちいちキーボードを[カナ]モードと[英数]モードに切り換える手間が省けます。

例）
```
10 MUSIC トレミフソラシト [RETURN]
>LIST 10 [RETURN]
10 MUSIC DトDレDミDフDソDラDシトレミフラソシ
         ↑
         └──画面エディターにより英数字を挿入する。
```

次に文字列および文字変数を使う方法について説明します。

> ＭＵＳＩＣステートメント中で用いる文字列および文字変数は音楽記号で構成しなければならない。文字列および文字変数については２８ページの文字変数の項を御覧ください。なお，文字列および文字変数を＋（プラス）の記号で結合するときは，連続して有効な音楽記号の結合に注意してください。
>
> 例）　ＭＵ〝Ｐ５トレＰ３ミミＰ７レ〞＋〝$\underset{\underset{\text{音の長さはＰ７。}}{\uparrow}}{\underline{\underline{\text{ト}}}}$Ｐ３レラソＵト〞

例102

```
 10  A$="トレミフソ"
 20  MUSIC  A$                  ----------- 例98 の左のプログラム
 30  MUSIC "Mソ"+A$             ----------- 例98 のプログラム
 40  MUSIC  RIGHT$(A$, 2)
 50  MUSIC  MID$(A$, 1, 2)
```

例103 「喜びの歌」(ベートーベン作曲）のプログラム。**例101** とくらべてみてください。

```
 10  PRINT  TAB (5); "ヨロコビノウタ"
 20  LET  A$="ミフソソフミレトトレミ"
 30  LET  B$="P6ミP3レP5レR"
 40  LET  C$="V4レレミトレP3ミフP5ミトレP3ミフ"
 50  LET  D$="P5ミレトレDソV5P7ミP5"
 60  LET  E$="P6レP3トP5トR"
 70  MUSIC  "MソT2V3ミ"+A$+B$ ----------- 例 101 の20行目
 80  MUSIC  "ミ"+A$+E$         ----------- 例 101 の30行目
 90  MUSIC  C$+D$+A$+E$        ----------- 例 101 の40行目
100  END
```

3.8 データの入出力に関するステートメント

ベーシックマスタージュニアに接続された入出力装置（たとえば，内蔵されているキーボード，別売のカセットテープレコーダー，プリンターなど）に，プログラム実行中に取り扱われるデータを入出力するためのステートメントで以下の4種類のステートメントがあります。

データの入出力に関するステートメント	OPEN	ある入出力装置とデータの送受ができるようにハード，ソフトを設定する。
	CLOSE	OPENで設定した状態を解除する。
	PRINT＃	設定した出力装置にデータを出力する。
	INPUT＃	設定した入力装置からデータを入力する。

これらのステートメントを使うと，たとえば計算結果や，データをプリンターに出力して，書類として保存することができます。

また，データをカセットテープに記録保存し，後日そのテープよりデータとして読み込むようなこともできます。

●OPENステートメント

OPENステートメントは，ある入出力装置とデータの送受ができるようにハード，ソフトを設定するステートメントです。

(1) OPENステートメントの一般形式

OPEN〈式1〉，〈式2〉，″ファイル名″，〈式3〉

ロジカルファイル番号（〈式1〉）	フィジカルデバイス番号（〈式2〉）	ファイル名	ユーザードライバー先頭番地（〈式3〉）
〈式1〉は〈式2〉，″ファイル名″,〈式3〉で設定する入出力装置を表わす。この，〈式1〉は1～15の範囲でユーザーが任意に定めることができる。	以下に定める装置を表わす。 1．画面 2．キーボード 3．カセット1OUT（SAVE） 4．カセット1IN（LOAD） 5．カセット2OUT（SAVE） 6．カセット2IN（LOAD） 7．～15．ユーザー定義デバイス	●外部記憶装置（カセットテープ等）に″　″でくくったファイル名をつけてデータを記録・再生する。 ●″　″内は6文字以内で，文字変数も使える。 ●データ再生のときはファイル名サーチができる。	●ユーザー定義デバイスをOPENするときは必ず必要。（フィジカルデバイス番号が7～15のとき） ●このとび先番地に，ユーザーが設定した入出力装置のドライバープログラムを作る。

(a) ロジカルファイル番号　　OPENステートメントで，データを入出力したい装置をこの番号に設定する。
一度設定すると，NEWコマンドを実行するか，CLOSEで解除しない限り，設定状態は維持される。

(b) フィジカルデバイス番号　　入出力装置そのものを表わす番号で，前頁の表に示すように1～6はすでに決まっており，7～15はユーザーが自由に定めることができる。

(c) ファイル名　　プログラムと同様，外部メモリーにデータを格納する際，ファイル名をつける。
これは，あってもなくてもよい。キーボードや画面をOPENするときは意味をなさない。

(d) ユーザードライバー先頭番地　　フィジカルデバイスとしてユーザーが装置を増設した際，その駆動ソフトウェア（ドライバー）が格納してある番地である。
フィジカル番号7～15を用いるときは必ず必要で，この番地(N)以降9バイトには次の命令が書かれていなければならない。

ユーザー装置内のジャンプテーブル

N　番地	$7E	（ジャンプ命令）	
N＋1番地	○○	ユーザー装置をOPENするサブルーチンの先頭番地	
N＋2番地	○○		
N＋3番地	$7E	（ジャンプ命令）	
N＋4番地	△△	ユーザー装置のドライバーサブルーチンの先頭番地	
N＋5番地	△△		
N＋6番地	$7E	（ジャンプ命令）	
N＋7番地	□□	ユーザー装置をCLOSEするサブルーチンの先頭番地	
N＋8番地	□□		

例）OPEN　2，3，〝ヨサン〟

意味　出力装置としてカセットテープをロジカルファイル番号2に設定し，データをカセットテープに格納する時は，ファイル名〝ヨサン〟をつける。

OPEN　1，7，〝ABC〟，$5000

意味　ユーザ定義の入出力装置をロジカルファイル番号1に設定し，データを格納するときはファイル名〝ABC〟をつける。その入出力装置のドライバープログラムは$5000番地以降に記載されている。

(2) システム内蔵のデバイスをオープンする場合

```
OPEN 〈式1〉, 〈式2〉
      ↑         ↑
      │         └─フィジカルデバイス番号 ………… (b)参照
      └─────────ロジカルファイル番号

OPEN 〈式1〉, 〈式2〉, "ファイル名"
                  "文字列 (6字列)" または文字変数または添字付文字変数
```

(a) 1度OPENしたロジカルファイルをCLOSEせずに2度以上OPENすると，ERROR6となり，最初にOPENしたロジカルファイルのみ有効となる。

(b) フィジカルデバイス番号

1．画面，2．キーボード

3．カセット1　OUT (SAVE)，4．カセット1　IN (LOAD)

5．カセット2　OUT，6．カセット2　IN (カセット2は存在しないので，5，6を設定すると，カセット1を設定したのと同じ結果になる。)

例)
```
10  OPEN  1, 3, "TEST1"
20  OPEN  2, 4, "TEST1"
```

意味　10行で "TEST1" というファイルをカセット1への書き込みモードでOPENし，20行で同じファイルをカセット1の読み込みモードでOPENする。

(3) ユーザー定義のデバイスをオープンする場合

```
OPEN 〈式1〉, 〈式2〉, 〈式3〉
      ↑         ↑
      │         └─フィジカルデバイス番号  7～15
      └─────────ロジカルファイル番号

OPEN 〈式1〉, 〈式2〉, "ファイル名", 〈式3〉
```

(a) 〈式3〉がない場合ERROR6となる。

(b) 〈式3〉には，$A00番地以降のBASICで使われていない番地を設定しなければならない。(グラフィック1を指定している場合は，$2200番地以降，グラフィック2を指定している場合は，$3A00番地以降に設定して下さい。)

(c) 〈式3〉で示された番地以降の9バイトに，ユーザー装置用ジャンプテーブルがないとERROR6となる。

例)　BASICプログラム　OPEN　1, 10, "TEST", $1000

機械語プログラム

$1000	7E110A	JMP $110A
$1003	7E○○○○	JMP $○○○○
$1006	7E△△△△	JMP $△△△△
$110A	番地以降にデバイスのオープンサブルーチンを作る。	

意味　"TEST" というファイルをユーザー定義のデバイス (フィジカルデバイス番号10) に対してOPENし，ロジカルファイル1に対応させる。このデバイスのオープンルーチンは，$1000番地のジャンプ命令で指定した$110A番地に先頭番地がある。

●PRINT＃ステートメント

設定した出力装置にデータを出力するステートメントです。

(1) ＰＲＩＮＴ ＃ ステートメントの一般形式

ＰＲＩＮＴ ＃ 〈式１〉，〈リスト〉

〈リスト〉──〈変数〉，〈式〉，〈定数〉，〈関数〉，〈その他〉

〈式１〉──ロジカルファイル番号

ＯＰＥＮステートメントで指定されたロジカルファイル（〈式１〉で示す）に〈リスト〉で示されるデータを出力する。

(a) 〈リスト〉は区切り記号を使って並べて書くことはできないので，１つのＰＲＩＮＴステートメントで出力できるデータは１個に制限される。

例） ＰＲＩＮＴ ＃１，Ａ　　ＯＰＥＮステートメントでロジカルファイル番号１と指定した装置に変数Ａの値をデータとして出力する。

(b) ユーザー定義のデバイスを指定した場合は，ユーザードライバー先頭番地＋３番地から＋５番地までに，ユーザー装置のドライバーサブルーチンにジャンプする命令がかかれていなければなりません。

(c) ユーザードライバーサブルーチンは，＄３Ｂ，＄３Ｃ番地に示されている番地から，＄３Ｄ，＄３Ｅ番地に示されている番地までのメモリーに格納されているデータを出力するように機械語を用いて作ります。

例） １０ ＯＰＥＮ １，３，"ＴＥＳＴ"　　カセット１をＳＡＶＥモードに設定
　　２０ ＬＥＴ Ｘ＝１．２３４５
　　３０ ＰＲＩＮＴ ＃１，Ｘ　　１０行でＯＰＥＮした装置にＸを出力

意味　カセット１をＯＵＴに設定し，Ｘの値をカセットにＳＡＶＥする。
　　　１０行の〈式１〉の１と，３０行の〈式１〉の１とが対応。

(2) プリンターに出力する場合

ＰＲＩＮＴ ＃，〈リスト〉

ＰＲＩＮＴ ＃，はプリンターに対するＯＰＥＮステートメントおよびロジカルファイル番号の指定を省略できるステートメントです。

(a) 〈リスト〉はＰＲＩＮＴステートメントと同様，区切り記号を使って並べて書くことができます。（行番号も含めて最大７９字まで）

例104

```
10 OPEN 1, 3, "TEST"………………カセット1をSAVEモード
20 LET X=1.2345
30 FOR I=1 TO 5
40 PRINT #1, I*2………………カセット1へ出力
50 PRINT#, I, I*2, I*3………………プリンターへ出力
60 PRINT I, I*2, I*3,………………画面へ出力
70 NEXT I
80 CLOSE 1………………10行のOPENを解除
99 END
```

例105 キーボードからのド，レ，ミ……の入力をそのまま音にするプログラムの例

```
10 OPEN 1, 10, $1A00………SIZEコマンドで使っていないメモリー
20 LET A$=INKEY$           領域を調べてから行なう。
30 MUSIC P3………………音の長さ，大きさ，音色等を指定できる。
40 PRINT #1, A$
50 PRINT A$
60 GOTO 20
```

機械語ジャンプテーブル

```
$1A00  7E  JMP $1A09
$1A01  1A
$1A02  09
$1A03  7E  JMP $F00C    MUSICルーチンへジャンプ
$1A04  F0
$1A05  0C
$1A06  7E  JMP $1A09
$1A07  1A
$1A08  09
$1A09  39  RTS          BASICの20行，50行に戻る。
```

●INPUT＃ステートメント

設定した入力装置からデータを入力するステートメントです。

```
INPUT ＃ 〈式1〉,〈変数〉
         ↑
         └──────ロジカルファイル番号
```

OPENステートメントで指定されたロジカルファイル（〈式1〉で示す）から〈変数〉にデータを入力する。

(a) ユーザー定義デバイスを指定した場合には，ユーザードライバー先頭番地＋3番地から＋5番地までに，ユーザー装置のドライバーサブルーチンにジャンプする命令が書かれていなければなりません。

(b) ユーザー装置のドライバーサブルーチンは，＄3B，＄3C番地に示されている番地以降に入力したデータを格納するように，機械語を用いて作ります。

例）
```
INPUT ＃2, X
        ↑
        └──────ロジカルファイル番号
```

意味　OPENステートメントでロジカルファイル番号2と指定した装置からデータを入力し，変数Xに代入する。

例）
```
10 OPEN 1, 4, "TEST"
20 INPUT ＃1, A$
30 PRINT A$
```

意味　カセット1より"TEST"というファイルより，1個の文字データを読み込み，A＄に代入し，画面にA＄を出力する。

●CLOSEステートメント

ＯＰＥＮステートメントで設定された入出力装置の状態を解除するステートメントです。

```
CLOSE 〈式〉
       └──ロジカルファイル番号
```

ＯＰＥＮステートメントでオープンされたロジカルファイルと入出力装置との関係を分離する。

ユーザー定義デバイスを指定した場合は，ユーザードライバー先頭番地＋6番地から＋8番地までに，ユーザー装置をＣＬＯＳＥするサブルーチンにジャンプする命令が書かれていなければならない。

例）
```
 10 OPEN n, m, "イロハ"
 20 OPEN k, 1
      :
 95 PRINT #n, A$ ………必ずOPENしてから使う。
      :
154 PRINT #k, X
      :
200 CLOSE n
220 CLOSE k
```

3.9 組み込み変数および組み込み定数

●CURSOR(組み込み変数)

表示画面は横32字，縦24行のマス目構造になっていて，ベーシックマスタージュニア内部では，この1つ1つのマス目にメモリーが割り当てられています。カーソルは，このマス目のどこに文字を画くかを示すステートメントです。つまりマス目に対応したメモリーの何番地かをカーソルが表示しているので，そのメモリーの中に表示したい文字を記憶させると自動的にそれが画面上に映し出されるようになっています。

CURSOR=〈式1〉，〈式2〉
　　　　↑横方向　↑縦方向

省略形　CUR
　または！

ます目の位置は，画面左上を原点として，横方向に0～31，縦方向に0～23の座標式表現で表わします。

CURSOR=〈数値変数〉※　　例）CURSOR=10，5は
　　　　　　　　　　　　　　CURSOR=$0A05と同じです。

〈数値変数〉=CURSOR

※ CURSORを一旦数値変数に退避し，再び呼びもどすことができます。

例106 横方向10，縦方向5の位置に記号#をプリントするプログラムです。

```
 10  CURSOR=10, 5
 20  PRINT "#"
 30  END
```

例107 カーソルをスィープさせながら図形を表示していくプログラムです。

```
 10  CLEAR
 20  FOR Y=0 TO 20 STEP 2
 30  FOR X=0 TO 22 STEP 2
 40  !=X, Y
 50  IF X>Y PRINT CHR$($95);
     :GOTO 70
 60  PRINT CHR$($94);
 70  NEXT X
 80  NEXT Y
300  END
```

●TIME（組み込み変数）

ベーシックマスタージュニアのタイマー機能と組み合わせ，１秒単位の時刻を表わす変数です。

ＴＩＭＥ＝〈式〉　タイマーカウンターに〈式〉の値を代入する。
〈数値変数〉＝ＴＩＭＥ　〈数値変数〉に１秒タイマーの値を代入する。
１．ＴＩＭＥは１秒単位で値が更新される。 ２．０≦ＴＩＭＥ≦３２７６７秒の範囲の時間を取り扱うことができる。 ３．ＭＵＳＩＣステートメント実行中はタイマーカウンターの更新が中断されます。

１/６０秒まで正確にしたいときは，まずＴＩＭＥ＝０とし，次にＰＲＩＮＴ　ＴＩＭＥ＋ＰＥＥＫ（＄Ｃ）/６０としてください。

例108　１０秒タイマー

```
１００　ＰＲＩＮＴ　"ＳＴＡＲＴ"
１１０　ＴＩＭＥ＝０……………………初期設定
１２０　ＩＦ　ＴＩＭＥ＜１０　ＧＯＴＯ　１２０……１０秒以下なら分岐。
１３０　ＰＲＩＮＴ　"ＥＮＤ"：ＥＮＤ
```

●PAI（組み込み定数）

円周率πを表わす定数です。

ＰＡＩ＝３．１４１５９２６５　（９桁）

例109　三角関数はラジアンで表わされます。

```
１０　ＣＬＥＡＲ
２０　ＦＯＲ　Ｒ＝０　ＴＯ　２＊ＰＡＩ　ＳＴＥＰ　ＰＡＩ／１０
３０　ＬＥＴ　Ｓ＝ＳＩＮ（Ｒ）
４０　ＰＲＩＮＴ　Ｓ
５０　ＮＥＸＴ　Ｒ
６０　ＥＮＤ
```

ヒント　１８時間タイマー

```
 １０　ＤＥＦ　ＦＮＴ＝ＴＩＭＥ＊１………ＴＩＭＥが３２７６７を超えた時，負の値にな
 ２０　ＴＩＭＥ＝０　　　　　　　　　　　らないように処理しています。
 ３０　Ｔ＝ＦＮＴ
 ４０　ＩＦ　ＦＮＴ＝Ｔ　ＧＯＴＯ　４０
 ５０　ＰＲＩＮＴ　ＦＮＴ
１００　ＧＯＴＯ　３０
```

3.10 関数／●関数について

ある引数に対してある数値または，文字列を一義的に定めることができるものを関数と呼びます。（引数が一見ないようなものもあるが，これらは，引数が固定か、あるいは引数に相当するものがある。）

1．分類

- 関数
  - 算術関数
    - 組み込み関数
      - ＳＩＮ（Ｘ），ＣＯＳ（Ｘ），ＴＡＮ（Ｘ）……三角関数
      - ＡＴＮ（Ｘ）……逆三角関数
      - ＥＸＰ（Ｘ），ＬＯＧ（Ｘ），ＳＱＲ（Ｘ）
      - ＡＢＳ（Ｘ），ＩＮＴ（Ｘ），ＲＮＤ（Ｘ），ＳＧＮ（Ｘ）
    - ユーザー定義関数　ＦＮｘ（Ｘ）
  - プリント文用関数…………ＴＡＢ（Ｘ），ＨＥＸ（Ｘ）
  - 特殊関数…………………ＰＥＥＫ（Ｘ）
  - 文字取扱い関数
    - 数値形　ＬＥＮ（Ｘ＄），ＡＳＣ（Ｘ＄），ＶＡＬ（Ｘ＄）
    - 文字形
      - ＬＥＦＴ＄（Ｘ＄，ｎ），ＲＩＧＨＴ＄（Ｘ＄，ｎ）
      - ＭＩＤＳ（Ｘ＄，ｎ，ｍ），ＳＴＲ＄（Ｘ）
      - ＣＨＲ＄（Ｘ），ＣＵＲＳＯＲ＄，ＩＮＫＥＹ＄
      - ＳＰＣ＄（Ｘ）

2．算術関数，文字取扱い関数の中の数値形（＄が後につかないもの）は，数値変数と同様に取り扱える。すなわち〈式〉の中にかける。

3．プリント文用関数は，ＰＲＩＮＴ文の中でしか使えない。

4．文字取扱い関数の中の文字形（＄があとにつくもの）は，文字変数と同様に取り扱える。

算術関数については94ページにかんたんな使用例を示します。文字取扱い関数については29ページ以降例をご参照ください。

ヒント　べき乗の計算について

べき乗はＬＯＧ（Ｘ）とＥＸＰ（Ｘ）関数を使って計算するため，計算する数字が極端に小さいか大きいとき，有効桁が小さくなりますが，それだけでなく計算に要する時間も長くなります。このためＡ∧３とか，Ｂ∧（１／２）の計算は，Ａ＊Ａ＊Ａとか，ＳＱＲ（Ｂ）で記述した方がより短い時間で処理できます。このことは次のプログラムで実験できます。

```
10 TIME=0
20 FOR A=1 TO 200
30 B=A^2
40 NEXT A
50 PRINT B,TIME
60 END
```

次に３０行を，次のように変更してください。

```
30 B=A*A
```

●算術関数

関数のキーワード	内容	引数の有効範囲	有効範囲をはずれたときの処理	備考
SIN(X)	正接 sinX (引数Xの単位はラジアン)	$\|X\| \leqq 4.29467299 \times 10^{9}$	$\|X\| \geqq 4.2946723 \times 10^{9}$ のとき SIN(X)=0	(注) 精度の高い計算を行なうときは $\|X\| \leqq 10^{2}$ に引数を小さくすることをおすすめします。(SIN(X)・COS(X)・TAN(X)共通)
COS(X)	余弦 cosX (引数Xの単位はラジアン)	$\|X\| \leqq 4.29467299 \times 10^{9}$	$\|X\| \geqq 4.2946723 \times 10^{9}$ のとき COS(X)=1	
TAN(X)	正接 tanX (引数Xの単位はラジアン)	$\|X\| \leqq 4.29467299 \times 10^{9}$	$\|X\| \geqq 4.2946723 \times 10^{9}$ のとき TAN(X)=0	
ATN(X)	逆正接 $\tan^{-1}X$ (ATN(X)の答の単位はラジアン)	$-1.935759908 \times 10^{9} \leqq X \leqq 1.908874355 \times 10^{9}$	OVERFLOW ERROR	$-\frac{\pi}{2}<ATN(X)<\frac{\pi}{2}$ $\|x\| \geqq 10^{10}$では $\frac{\pi}{2}$となる。
EXP(X)	自然対数の底eのべき乗 e^{X}	$\|X\| \leqq 88.0296918$	X>88.0296918のとき OVERFLOW ERROR X<−88.0296918のとき EXP(X)=0	
LOG(X)	自然対数 $\log_e X$	$0 \leqq X \leqq 1.70141183 \times 10^{38}$	X≦0のとき ERROR 5を表示する。 $X>1.70141183 \times 10^{38}$ のときOVERFLOW ERROR	
SQR(X)	平方根 $\sqrt{X}$	$0 \leqq X \leqq 1.70141183 \times 10^{38}$	X<0のとき ERROR 5を表示する。 $X>1.70141183 \times 10^{38}$ のときOVERFLOW ERROR	
ABS(X)	Xの絶対値 $\|X\|$	扱える数の全範囲	OVERFLOW ERROR	
INT(X)	Xをこえない最大の整数	同　上	OVERFLOW ERROR	
SGN(X)	Xの符号を判定。 X<0ときSGN(X)=1 X=0 〃 SGN(X)=0 X>0 〃 SGN(X)=1	同　上	OVERFLOW ERROR	
RND(X)	0〜Xまでの乱数を発生	$0<X<1.70141183 \times 10^{38}$	X≦0のときRND(X)=0 $X>1.70141183 \times 10^{38}$ のときOVERFLOW ERROR	
FNx(X)	ユーザーが式を組み合わせて作った関数	詳細は69ページ参照		

例110　SIN (X), COS (X)

```
10 CLEAR
20 LET X=22
30 FOR I=1 TO 5
40 FOR H=0 TO PAI*2 STEP .05
50 LET X=X-.03
60 PLOT COS (H) *X+32, SIN (H) *X+22, 1
70 NEXT H
80 NEXT I
90 END
```

例111　ATN (X)　単位はラジアン（角度）で出てきます。

```
10 CLEAR
20 INPUT "ザヒョウ X=", X
30 INPUT "ザヒョウ Y=", Y
40 LET C=ATN (Y/X)
50 LET C=C*180/PAI ………… 2πrad=360°
60 LET L=SQR (X^2+Y^2)
70 PRINT "ザヒョウ"; X; ", "; Y; " ノ"
80 PRINT "キョクカクド ハ "; C; " ド"
90 PRINT "キョリ ハ"; L
100 END
```

例112　LOG (X) の底はe（=2.71828……）常用対数の底は10。

```
10 DEF FNL (X) =LOG (X) /LOG (10) ……log_a b = log_b / log_a
20 INPUT X
30 PRINT FNL (X)
40 GOTO 20
```

例113　EXP (LOG(A)) =Aです。この計算はべき乗でやった方が早い。

```
10 CLEAR
20 INPUT "A=", A
30 INPUT "B=", B
40 LET C=EXP (LOG (A) *B)
50 PRINT A; " ノ "; B; " ジョウ ハ "; C
60 END
```

例114 ABS（X），SGN（X），RND（X）

```
 10 CLEAR
 20 LET A=0:LET B=0
 30 FOR K=1 TO 20
 40 LET R=RND(100)-50
 50 PRINT R
 60 IF R=ABS(R) THEN LET A=A+1:GOTO 80……正の条件
 70 LET B=B+1…………………………………………負をカウント
 80 NEXT K
 90 PRINT "プラス ノ カズ ハ ";A
100 PRINT "マイナス ノ カズ ハ ";B
110 END
```

次に，60行を次のように書きなおしてみてください。

```
 60 IF SGN(R)>=0 LET A=A+1:GOTO 80
```

例115 INT（X）　インチは12進法です。

```
 10 CLEAR
 20 INPUT "ナンメートル デスカ",M
 30 LET F1=M/.3048
 40 LET F2=INT(F1):LET F3=F1-F2……F3は小数部
 50 LET I1=F3*12
 60 LET I2=INT(I1)…………………………1インチ未満切り捨て
 70 CLEAR
 80 PRINT M;"メートル ハ";
 90 IF F2=0 THEN GOTO 110
100 PRINT F2;"フィート";
110 IF I2=0 THEN GOTO 130
120 PRINT I2;"インチ"
130 END
```

例116 FNx（X）　下2桁は60進で秒，3桁以上は10進の分。電源ONまたは，TIME=0を実行してから今までの時間です。

```
1000 DEF FNT(T)=INT(T/60)*100
       +T-INT(T/60)*60……分も60進にするには？宿題です。
1010 LET T=TIME
1020 CURSOR=25,0
1030 PRINT FNT(T)
1040 GOTO 1000
```

●文字取り扱い関数

関数のキーワード	例	内容
VAL(X$)	10 A=B−VAL (X$)	X$で表わされる文字列を数値に変換する。引数X$は文字型であり，その文字列は+，−，・，Eと数字のみで構成されていること。 文字型 → 数値型。
STR$(X)	10 A$=〝セイレキ″ +STR$ (1982)	Xで表わされる数値を文字列に変換する。Xは数値変数，定数など数値を表わすものであること。 数値型 → 文字型
ASC(X$)	10 X$=〝ABC″ 20 PRINT ASC (X$) RUN 65	X$で表わされる文字列の先頭文字の文字コード（47ページの文字図形の表示コード表のコード）を10進数に変換する。 ASCIIの意。 文字型 → 数値型
LEN(X$)	10 X$=〝イロハ″ 20 PRINT LEN (X$) RUN 3	X$で表わされる文字列の長さ（文字の数）を表わす。LENGTHの意。 文字型 → 数値型
LEFT$ (X$, N)	10 X$=〝1982ネン1ガツ″ 20 B$=LEFT$ (X$, 6) 30 PRINT B$	X$で表わされる文字列の左端からN個の文字列を取り出す。
RIGHT$ (X$, N)	10 X$=〝1982ネン1ガツ″ 20 B$=RIGHT$ (X$, 4) 30 PRINT B$	X$で表わされる文字列の右端からN個の文字列を取り出す。
MID$ (X$, N, M)	10 X$=〝1982ネン1ガツ″ 20 B$=RIGHT$ (X$, 4) 30 C$=MID$ (X$, 3, 4) 40 PRINT B$, C$ RUN 1ガツ 82ネン	X$で表わされる文字列の左からN番目の文字からM個の文字列を取り出す。Mを省略するとN番目から最終の文字までとなる。
SPC$(X)	10 X$=〝1982ネン″ 20 PRINT SPC$ (5)+X$ RUN ␣␣␣␣␣1982ネン	X個のスペースから成り立つ文字列を表わす。
CHR$ (X,Y,…)	10 PRINT CHR$ ($31, $39, $38, $32, $C8, $DD) RUN 1979ネン	文字コード（47ページの文字図形の表示コード）X，Y，……の文字を表わす。
CURSOR$ 省略形 CUR$, 又は/$	10 CUR=5, 5 : ?〝A″ : CUR=5, 5 20 A$=/$: /=10, 10 30 PRINT A$	画面上のCURSOR位置にある文字を表わす。引数はないがCURSORの位置が引数相当。
INKEY$	10 A$=INKEY$ 20 PRINT A$ 30 GOTO 10	おされたキーの文字を表わす。 引数はないがおしたKEYが引数相当。 キーがおされてないときは$00が入っています。

3.11 プログラムの記録・再生

作ったプログラムはカセットテープレコーダーを使ってテープに記録し保存することができます。また，カセットテープに記録したプログラムを，再びベーシックマスタージュニアのメモリーの中に移してプログラムを再生することができます。

カセットテープにプログラムを記録する場合，記録するプログラムにファイル名をつけて記録することができ，また，プログラムを再生するときもそのファイル名を使って，容易に目的のプログラムを再生することができます。

ファイル名による呼び出しとは，例えば，1本のテープに〝A〟，〝B〟，〝C〟と名付けた3種類のプログラムを順に記録したとすると，次に〝B〟のプログラムを呼び出したいとき，「Bのプログラムを呼び出せ」と指示を与え，カセットテープを走行させると，〝B〟の前後に入っている〝A〟や〝C〟は呼び出さず，〝B〟のみを呼び出して，メモリー内に移すことを言います。

●記録・SAVEコマンド

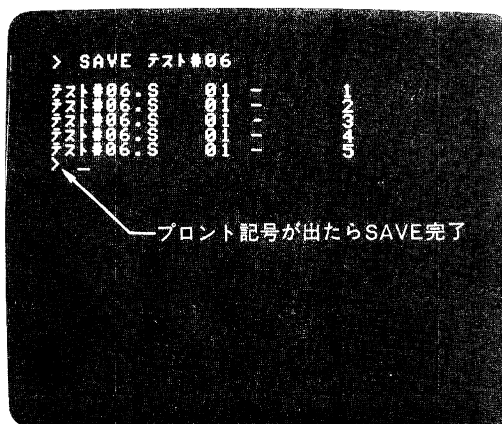
ＳＡＶＥコマンドにより，テープレコーダーにプログラムを記録する。

ＳＡＶＥ　〈ファイル名〉 [RETURN] →例　ＳＡＶＥ　ゲームＡ [RETURN]
（ゲームＡ →ファイル名）

〈ファイル名〉は最大6字までの英大文字，カナ，記号が使えます。ご自由にファイル名をつけて下さって結構ですが，後での検索に便利なように，プログラムの内容を端的に表すファイル名をつけることをおすすめします。[RETURN]を押すまえに，テープレコーダーの操作ボタンを操作して録音状態にし，テープをスタートしてから[RETURN]を押します。〔テープの始めにあるリーダテープ部（記録できない部分で普通色が異る。）の部分からＳＡＶＥしないようご注意下さい。〕

[RETURN]を押すと順次，図のように表示されます。ファイル名のあとに，Ｓと表示されるのは，プログラムがベーシックのものであることを示します。

記録の進行に従い，ファイル名のあとにブロック長を示す０１という番号が，さらにそのあとにブロックの番号が１０進数表現で表示されます。記録が終るとファイル名の下にプロンプト記号が表示されます。



記録が終ったらテープレコーダーを停止します。

1．ファイル名は最大6文字までで，7文字以上のファイル名を指定すると，エラー（ＥＲＲＯＲ　3）を生じます。

2．テープレコーダーの録音レベルが自動調節されない場合，レベルオーバーにならないように注意してください。次に説明するＶＥＲＩＦＹコマンドにより，正確に記録されているかどうかを確認し，適正な録音レベルを決めてください。

3．テープレコーダーの電池が消耗していたり，回転速度が不安定な場合，正確に記録できないことがあります。

4．ピッチコントロール(又はスピードコントロール)等のテープスピードが可変できる機能がついているものでは，必ずテープスピードは標準の速さを選ぶようにして下さい。異ったテープスピードでＳＡＶＥや以下に述べますＶＥＲＩＦＹ，ＬＯＡＤを行いますとエラーの原因となりますし，他のテープレコーダーを使用する場合テープの互換性が失なわれてエラーの原因となりますのでご注意下さい。

5．ＳＡＶＥ，ＶＥＲＩＦＹ，ＬＯＡＤ，ＭＥＲＧＥを通じカセットケーブルの２本の線（各々「出力端子へ」と「入力端子へ」と表示されている線）は，間違いのないように両方ともテープレコーダーに接続してください。

6．トーンコントロール(音質調整)が付いているテープレコーダーはこれを高音の出るように設定して下さい。

●確認・VERIFYコマンド

ＳＡＶＥコマンドにより記録したプログラムが正しく記録できているかどうかを確認するのがＶＥＲＩＦＹコマンドです。

ＶＥＲＩＦＹコマンドは，ベーシックマスタージュニア内のメモリーにあるプログラムと，記録したテープのプログラムとの比較確認をするものです。

テープを記録開始点のすこし前まで巻き戻しておきます。

ＶＥＲＩＦＹ　〈ファイル名〉 RETURN

〈ファイル名〉はＳＡＶＥコマンドで記録したときと同じものをキー入力します。RETURN を押した後，テープレコーダーを走行させ，テープを再生します。

```
> VERIFY テスト#06
テスト#06.S     01
テスト#06.S          -     1
テスト#06.S     01
テスト#06.S          -     2
テスト#06.S     01
テスト#06.S          - ERROR
テスト#06.S     01
テスト#06.S          -     4
テスト#06.S     01
テスト#06.S
> _
```

ブロック３にメモリー内容のエラー。

テープ走行に従って右図のような表示が現われます。

正しく記録され，かつ正しく再生されれば，ファイル名のあとにブロック長を示す０１という番号が，さらにつぎの行にファイル名とブロック番号が表示されます。

このコマンドのエラーには，

(1) ─ ＥＲＲＯＲ

(2) ＊ＥＲＲＯＲ　の２種類の違ったエラー表示があります。

─　ＥＲＲＯＲの表示の場合は，ベーシックマスタージュニア内のメモリーにあるプログラムと，テープに記録されているプログラムに違いがある場合です。

また，＊ＥＲＲＯＲの場合は，誤って記録，または再生された場合です。

＊エラーが表示された場合，エラーの原因としては，録音レベルが大きすぎ，歪が生じている。再生レベルが大きすぎ，歪が生じている。回転にムラがある。などが考えられます。次の対策を行ってみてください。

1. 再びVERIFYを行う。
2. テープレコーダーの出力レベル(再生レベル)を可変して再度VERIFYを行う。
3. SAVEをやり直してみる。
4. テープを取り替えてSAVE, VERIFYを行う。
5. テープレコーダーのヘッドをそうじしてみる。テープレコーダーをとりかえてみる。
6. テープレコーダーを電池駆動している場合は，電池が消耗してテープスピードが低下していないか調べてみる。
7. トーンコントロール（音質調整）が付いているものではこれを変化させ再度VERIFYを行う。

●再生格納・LOAD／MERGEコマンド

すでにテープに記録してあるプログラムをベーシックマスタージュニアのメモリーに格納するコマンドです。このとき，すでにメモリーに格納されていたプログラムを消去してからプログラムの格納を行うのがLOADコマンドで，消去を行なわずにプログラムを追加しながら格納するのがMERGEコマンドです。

目的のプログラムが記録されていると思われる個所の前まで，テープカウンターなどを参照して，テープを巻き戻し（早送りし）ておきます。

LOAD　〈ファイル名〉 RETURN

MERGE　〈ファイル名〉 RETURN

RETURN を押したのち，テープを再生走行させます。再生出力レベルが適切でないとエラーしますので，VERIFYコマンドのとき，正しく動作したボリューム位置に設定してください。

```
> LOAD テスト#06
テスト#06.S    01
テスト#06.S       -    1
テスト#06.S    01
テスト#06.S       -    2
テスト#06.S    01
テスト#06.S       -    3
テスト#06.S    01
テスト#06.S       -    4
テスト#06.S    01
テスト#06.S
>
> MERGE テスト#07
テスト#07.S    01
テスト#07.S       -    1
テスト#07.S    01
テスト#07.S       -    2
テスト#07.S    01
テスト#07.S       -    3
テスト#07.S    01
テスト#07.S
> _
```

ブロック長を示す記号
ブロック番号
プロント記号が出たらLOAD完了

正しく再生格納されると右図のようにファイル名のあとにブロック長を示す01という番号が，さらに次の行にファイル名とブロック番号が表示されます。

MERGEの場合，すでにベーシックマスタージュニアのメモリーに格納されていた行番号と同一の行番号を持ったプログラムをテープから再生すると，すでにメモリーに格納されていたその行番号の1行は消去され，テープから再生される行番号の1行が新しくメモリーに格納されます。※

何らかの原因でエラーが発生した場合，ピーッという警告音とともにVERIFYと同様のエラーが表示されます。

※前に入っているプログラムを消去したくない場合はRESEQを使用するなどして同一の行番号が存在しないようにしておく必要があります。

エラー表示のうち，＊ＥＲＲＯＲは誤って再生された場合ですので，ＳＡＶＥの項で述べたと同様な対策を行なってみてください。なお別のテープからＬＯＡＤしてみて，エラーが出なければ，エラーの出たプログラムはＳＡＶＥが正しく行なわれなかったかあるいは正しくＳＡＶＥした後何らかの原因でテープに傷がついたりして記録内容がおかしくなったものと考えられます。

ご注意

カセットケーブル中のリモート端子へ接続する黒いケーブルは，ＬＯＡＤなどの動作中におけるテープの走行を制御するためのものです。次のようにお使いください。

1．通常のプログラムのＳＡＶＥ，ＬＯＡＤなどの場合は，特にテープの走行を制御しなくても正常な動作を行ないますので，リモート端子に本ケーブルを接続しないでご使用ください。(接続しない方がテープの早送り，巻戻しなどが容易に行なえます。)
2．特に長いプログラムのＬＯＡＤ，ＭＥＲＧＥ中で　ＥＲＲＯＲ　２　が生じる場合，本ケーブルをリモート端子に接続しご使用ください。
3．上記２の場合，テープの早送り，巻戻しは本ケーブルをリモート端子から抜くか，または単にＶＥＲＩＦＹ [RETURN] とキー入力した後に行なうことができます。(ＶＥＲＩＦＹ [RETURN] 後，早送り，巻戻しが終ったら [BREAK RESET] キーを押すとプロンプトマークが出て，次の入力待ちになります。)

ヒント　プログラムのＬＯＡＤ，ＭＥＲＧＥ中にエラー表示が出て，プロンプトマークが表示されたら。

少し巻き戻して途中からＭＥＲＧＥコマンドを実行して，前にエラーとなったブロックがエラーなしで表示されればＯＫ。

先頭からＬＯＡＤしなおす必要はありません。

再度同一のエラー表示が出て，プロンプトマークが表示されてしまったときは，残念ですがその２つのプログラムをＭＥＲＧＥすることができません。めんどうでも残りのプログラムをキーボードから入力してください。

例
```
テスト＃０１．Ｓ　０１
テスト＃０１．Ｓ　０１　－１
テスト＃０１．Ｓ　　＊ＥＲＲＯＲ
ＳＹＮＴＡＸ　ＥＲＲＯＲ
>ＭＥＲＧＥ　テスト＃０１
テスト＃０１．Ｓ　０１
テスト＃０１．Ｓ　０２　－２
```

(プログラムのＬＯＡＤ，ＭＥＲＧＥを始めたとき，エラー表示をせずプロンプトマークが表示されたら，テープを少し巻き戻してから再ＬＯＡＤ，ＭＥＲＧＥをしなおしてください。)

●BASICで機械語のプログラムをLOADする方法

●通常の場合，

ＢＡＳＩＣのプログラムのみをテープに記録，再生するときはＳＡＶＥ，ＬＯＡＤコマンドを，機械語のプログラムのときはモニターのＰ，Ｌコマンドを使います。

●以上の使い方の他に，

ＢＡＳＩＣと機械語の両方のプログラムをＬＯＡＤしたい場合，ＢＡＳＩＣのプログラムはもちろん，機械語のプログラムもＢＡＳＩＣのコマンドでＬＯＡＤすることができます。なお，テープに記録するときは，ＢＡＳＩＣはＳＡＶＥコマンド，機械語はモニターのＰコマンドで各々別々に記録しなければなりません。

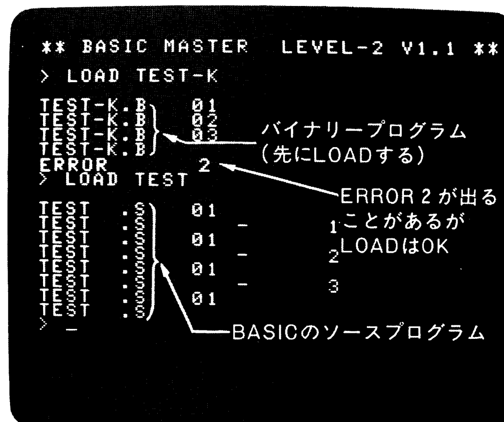
●操作法

機械語のプログラムをファイル名〝ＴＥＳＴ－Ｋ〟で，またＢＡＳＩＣのプログラムをファイル名〝ＴＥＳＴ〟でテープに記録してあるものと仮定します。

ａ．ＬＯＡＤ　ＴＥＳＴ－Ｋ RETURN とキー入力し，ＢＡＳＩＣで機械語のプログラムをＬＯＡＤします。

ＴＥＳＴ－Ｋ．Ｂ　01
↑ 機械語のプログラムの場合はＢとなる。

〈注〉ＢＡＳＩＣと機械語とではプログラムの記録・再生の方式が異なるため，右図のように機械語のＬＯＡＤ終了後ＥＲＲＯＲ　２が表示されますが，メモリーには正しく格納されています。



ｂ．次にＬＯＡＤ　ＴＥＳＴ RETURN とキー入力し，ＢＡＳＩＣのプログラムをＬＯＡＤします。

ＴＥＳＴ　．Ｓ　01
↑ ＢＡＳＩＣのプログラムの場合はＳとなる。

以上ａ，ｂの操作を行なえば，ＢＡＳＩＣのＬＯＡＤコマンドでＢＡＳＩＣと機械語のプログラムがＬＯＡＤできます。なお，ＬＯＡＤの順番は必ず機械語のプログラムを先に行なってください。

プログラム作成上のメモ

ＢＡＳＩＣと機械語のファイル名を同一にしておくと，同一ファイル名でＬＯＡＤできます。このとき，ＢＡＳＩＣのプログラムか機械語のプログラムかは，ファイル名につづく，．Ｓ，．Ｂで区別できますが，ＬＯＡＤ〈ファイル名〉の操作は２回やらなければなりません。なお，プログラムは機械語，ＢＡＳＩＣの順に記録しておくことが必要です。

● DIMの領域をデータとしてカセットテープにSAVE，LOADする方法

ベーシックマスタージュニアのプログラムの中で，ＤＩＭステートメントで宣言した領域のデータのみをデータとしてカセットテープにＳＡＶＥあるいは，ＬＯＡＤすることができます。

1.SAVE

配列を使ったプログラムの中には，１回ＲＵＮさせることによって初めて添字付変数に定数が代入されるものがあります。（例，49ページ，**例 50**）このようなプログラムでは，ＲＵＮしないと，定数がメモリーの配列エリアに格納されません。

(a) ベーシックマスタージュニアでは１つの数値を５バイトのデータとして，メモリーの最終番地（ＲＡＭＥＮＤ）より順に格納してゆくので，次のようなプログラムをメインプログラムのあとに作ってください。

```
   10  DIM A(50), B(3, 2)    } メインプログラム
   20   :                      配列宣言はメインプログラムの最
        :                      初に行なう。
  999  END
10000  REM *** SAVE ***
10010  S9=PEEK(8)*256+PEEK(9)
10020  S9=S9-(5*50+2)-(5*3*2+4)
              └A(50)のメモリー  └B(3, 2)のメモリー……次頁〈注意〉参照
10025  CALL $F682
10030  POKE $3B, S9/256
10040  POKE $3C, S9-INT(S9/256)*256
10050  POKE $3D, PEEK(8)
10060  POKE $3E, PEEK(9)
10080  CALL $31
10090  END
```

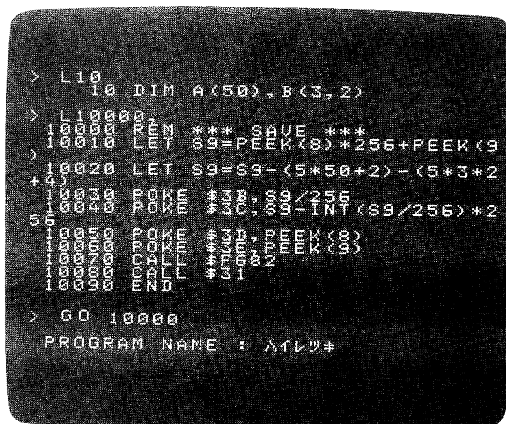
(b) ＧＯ 10000（追加プログラムの先頭）RETURN を実行すると右図のように

ＰＲＯＧＲＡＭ ＮＡＭＥ ： ＊

と入力要求してきますので，データのファイル名をキー入力してください。

このとき，６文字打ち終るとただちにＳＡＶＥを開始するので，ファイル名をキー入力する前にテープを録音走行させてください。

なお，この追加プログラムを実行する前に，メインプログラムがＲＵＮされ，実数が配列エリアに格納されている必要があります。



(c) 記録が開始されると，右図のように表示され，ブロック番号が表示されます。

プロンプト記号が出たらＳＡＶＥ完了です。次に説明するＶＥＲＩＦＹプログラムにより正しく記録されたかを確認してください。

```
   10 DIM A(50),B(3,2)

> L10000,
 10000 REM *** SAVE ***
 10010 LET S9=PEEK(8)*256+PEEK(9
)
 10020 LET S9=S9-(5*50+2)-(5*3*2
+4)
 10030 POKE $3B,S9/256
 10040 POKE $3C,S9-INT(S9/256)*2
56
 10050 POKE $3D,PEEK(8)
 10060 POKE $3E,PEEK(9)
 10070 CALL $F682
 10080 CALL $31
 10090 END

> RUN 10000

 PROGRAM NAME : ﾊｲﾚﾂｽｳ

ﾊｲﾚﾂｽｳ.B   01
ﾊｲﾚﾂｽｳ.B   02
> _
```

〈注意〉 一般的に配列宣言によって確保されるメモリーの大きさは次のように表わすことができますので，前頁１００２０行の〰〰部の値は下記により定めてください。

１）１次元の配列　　　Ａ（Ｎ）‥‥‥‥５×Ｎ＋２
２）２次元の配列　　　Ａ（Ｎ，Ｍ）‥‥５×Ｎ×Ｍ＋４
３）１次元文字配列　　Ａ＄（Ｎ）‥‥‥３３＊Ｎ＋２

１≦Ｎ，Ｍ≦２５５

2.LOAD (VERIFY)

(a) ＳＡＶＥとは逆に，あらかじめ確保したメモリー領域にカセットテープからデータを入力するには，次のようなプログラムをつくってください。

ＶＥＲＩＦＹのときは＄５Ｃ番地に１を入力します。

```
   10 DIM A(50), B(3,2)
   20 GOSUB 10100
   30 END
10100 REM ** LOAD/VERIFY **
10110 POKE $5C, 0 ←―――― VERIFYのときは1を入れる。
10120 CALL $F682 ‥‥‥‥‥ プログラム名入力サブルーチンをコール
10130 CALL $2E ‥‥‥‥‥‥ データブロック入力サブルーチンをコール
10140 RETURN
```

(b) ＲＵＮを行なうと，ファイル名を要求してくるので，ファイル名をキー入力すると，右図のように，ＬＯＡＤまたはＶＥＲＩＦＹできます。

```
> L10100,
 10100 REM *** LOAD/VERIFY ***
 10110 POKE $5C,1
 10120 CALL $F682
 10130 CALL $2E
 10140 END

> RUN 10100

 PROGRAM NAME : ﾊｲﾚﾂｽｳ

ﾊｲﾚﾂｽｳ.B   01
ﾊｲﾚﾂｽｳ.B   02
ﾊｲﾚﾂｽｳ.B
> _
```

注意事項

１，２で説明した方法では，６文字キー入力するとすぐ実行が開始されます。ミスキー入力などを防ぐため，ＣＡＬＬ　＄Ｆ６８２ステートメントのあとに，ＳＴＯＰステートメントを挿入すると，ファイル名，テープレコーダーの準備などを確認して，ＣＯＮＴ［RETURN］で実行することができます。

●データの入出力

ベーシックマスタージュニアで，特に大量のデータをカセットテープに記録，再生したいときに使用します。

1. プログラムの作り方

(a)データ出力の場合

計算した結果やデータをカセットテープに記録保存する場合，以下に示した3種類のステートメントを使用してプログラムを作成してください。

〈行番号〉　ＯＰＥＮ　〈式１〉，〈式２〉，〝ファイル名〟，＄Ｅ２００
データの出力モードを設定し，出力ファイル名を指定する。
〈行番号〉　ＰＲＩＮＴ　＃〈式１〉，〈変数等〉
データをテープに出力する。
〈行番号〉　ＣＬＯＳＥ　〈式１〉
データの出力モードを解除する。

〈式1〉：ロジカルファイル番号と呼び，ＯＰＥＮステートメントで指定した入出力装置の識別番号であり，1～15の任意の値を指定できる。

〈式2〉：フィジカルデバイス番号と呼び，装置固有の識別番号であるが本使用方法内では7～15の任意の値を選択できる。

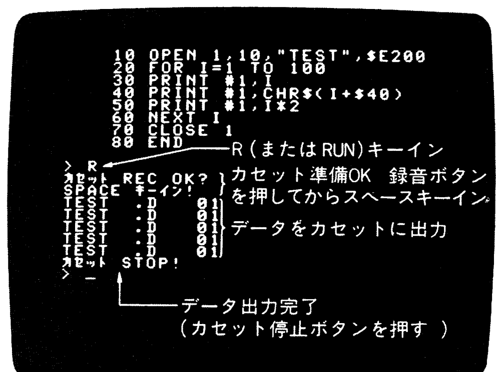
〝ファイル名〟：カセットテープにファイル名をつけてデータを記録できる。名前は〝　〟内に6文字以内の英数字，カナ，記号を使って指定できる。（文字変数も可能）

〈変数等〉：データ値で〈変数〉,〈式〉,〈定数〉,〈関数〉等の表現ができる。

注）〈変数等〉は区切り記号を使って並べて書くことはできない。
1つのＰＲＩＮＴ＃ステートメントで出力できるデータは1個である。

例117 データ出力プログラムおよび操作例

```
10  OPEN 1,10,"TEST",$E200
20  FOR I=1 TO 100
30  PRINT #1, I
40  PRINT #1,CHR$(I+$40)
50  PRINT #1, I*2
60  NEXT I
70  CLOSE 1
80  END
```



(b) データ入力の場合

あらかじめカセットテープに記録保存しているデータを再生する場合は，以下に示した3種類のステートメントを使用してプログラムを作成してください。

> 〈行番号〉　OPEN　〈式1〉，〈式2〉，〝ファイル名〟，＄E209
>
> 　データの入力モードを設定し，入力ファイルを指定する。
>
> 〈行番号〉　INPUT　#〈式1〉，〈変数等〉
>
> 　テープのデータを入力する。
>
> 〈行番号〉　CLOSE　〈式1〉
>
> 　データの入力モードを解除する。

〈式1〉：ロジカルファイル番号と呼び，OPENステートメントで指定した入出力装置の識別番号であり，1～15の任意の値を選択できる。

〈式2〉：フィジカルデバイス番号と呼び，装置固有の識別番号であるが，本使用方法内では，7～15の任意の値を選択できる。

〝ファイル名〟：カセットテープに記録しているデータのファイル名である。

〈変数等〉：データを格納する変数名，配列名で，カンマ（，）を使って並べて書くことができる。

例118　**例117** で出力したデータを入力するプログラムと操作

```
10 DIM A(100),B$(100),C(100)………データを格納するための配列エリア
20 OPEN 1,10,"TEST",$E209……"TEST"というファイル名の
                                データを読み込みモードでオープン。
30 FOR I=1 TO 100
40 INPUT #1,A(I),B$(I),C(I)……入力の場合は(,)で区切って書ける。
50 NEXT I
60 CLOSE 1
70 END
```

R（またはRUN）RETURN を入力すると右図のように画面に表示されます。カセットを再生状態にしてスタートしてください。

（右図では表示の関係上、行番号10のプログラムがでていません。）

```
20 OPEN 1,10,"TEST",$E209
30 FOR I=1 TO 100
40 INPUT #1,A(I),B$(I),C(I)
50 NEXT I
60 CLOSE 1
70 END
> R ←──R（またはRUN）キーイン
ｶｾｯﾄ PLAY!←──カセットからのデータ待ち
TEST  .D   01 ┐（カセット再生スタート）
TEST  .D      │
TEST  .D   01 │
TEST  .D      │
TEST  .D   01 ├データ入力（ファイル名表示）
TEST  .D      │
TEST  .D   01 │
TEST  .D      │
TEST  .D   01 ┘
TEST  .D
ｶｾｯﾄ STOP!←──データ入力完了
> _              （カセット停止ボタンを押す）
```

注）大量のデータを入力する場合は，データ入力の間に他の仕事をしようとすると，処理内容によっては，順番通りデータを読みこまないことがあります。そのときは，以下の例に示したように一度データを添字付変数（配列）に読み込んでから，必要とする変数に置き換えてください。

例119 データ入力を誤る例

```
 10  OPEN 1,10,"TEST",$E209 ………… カセットテープよりファイル名が"TEST"
                                    のデータを入力に設定
 20  INPUT #1, N ……………………………… データの個数N(組)を読み込む
 30  FOR I=1 TO N
 40  INPUT #1,X,Y ………………………… 2個のデータをX，Yに代入する
 50  LET Z=SIN(X)*2+COS(Y)*2 ……… そのX，Yをもとに計算→Z ┐ データ入力の間に別
 60  PRINT I, Z ……………………………… 計算結果を画面に表示     ┘ の仕事をしている。
 70  NEXT I
 80  CLOSE 1 ………………………………… カセット入力モードを解除
900  END
```

例120 **例119** を修正する

```
 10  DIM X,(100),Y(100) ……………… 読み込んだデータを一時格納する場所を確保する
 20  OPEN 1,10,"TEST",$E209
 30  INPUT #1,N
 40  FOR I=1 TO N ………………… ┐ 読み込みだけの処理
 50  INPUT #1,X(I),Y(I)         │ 読み込んだデータを一時X(I),Y(I)に格納
 60  NEXT I …………………………… ┘
 70  CLOSE 1 ………………………… カセット入力モードを解除
 80  FOR I=1 TO N ………………… ┐
 90  LET Z=SIN(X(I))*2+COS(Y(I))*2 │ 計算，表示のループを別に作る
100  PRINT I,Z                  │
110  NEXT I …………………………… ┘
900  END
```

3.12 プリンターの使い方

日立ドットインパクトプリンターMP－1041またはMP－1045を使って，プログラムのリストや，データを印字することができます。ご使用にあたっては，あらかじめMP－1041またはMP－1045の取扱説明書をご覧ください。

ここでは，MP－1041の使い方を説明します。

1. プリンターへのプログラムリスト出力

作成したプログラムのリストを出力したい場合は，1行あたりの文字数により，次のように実行してください。

a．132字／行モード

CALL　$E02B RETURN

b．80字／行モード（標準状態）

LIST#　（又はL#） RETURN

c．40字／行モード

CALL　$E02D RETURN

2. プリンターへのデータ出力

プログラムを実行する際，計算結果やデータをプリンターに出力したいときは，右表にしたがってプログラムを作成してください。また，印字される文字，図形，コントロールコードは，MP－1041の伝送データコードに従います。

文字数/行のモード	プログラム
132字モード	OPEN〈式1〉，〈式2〉，$E009 PRINT #〈式1〉，〈変数等〉 CLOSE〈式1〉
80字モード	OPEN〈式1〉，〈式2〉，$E000 PRINT #〈式1〉，〈変数等〉 CLOSE〈式1〉
40字モード	OPEN〈式1〉，〈式2〉，$E012 PRINT #〈式1〉，〈変数等〉 CLOSE〈式1〉

3. 画面文字のプリンター出力

画面上に出ている文字をプリンターにそのまま出力することができます。

a．ダイレクト実行の場合

CALL　$E021 RETURN

b．プログラム中で行う場合

```
10 REM**ガメンコピー**
20 FOR I=$21 TO $7A
30 PRINT CHR$(I),
40 NEXT I
50 CALL $E021
60 END
```

注1．この命令でプリンターに出力できるのは，下表の文字と記号だけです。空欄にはスペースが出力され何も印字しません。

注2．この命令で高解像度グラフィックの画面は出力できません。

出力できるキャラクタコード

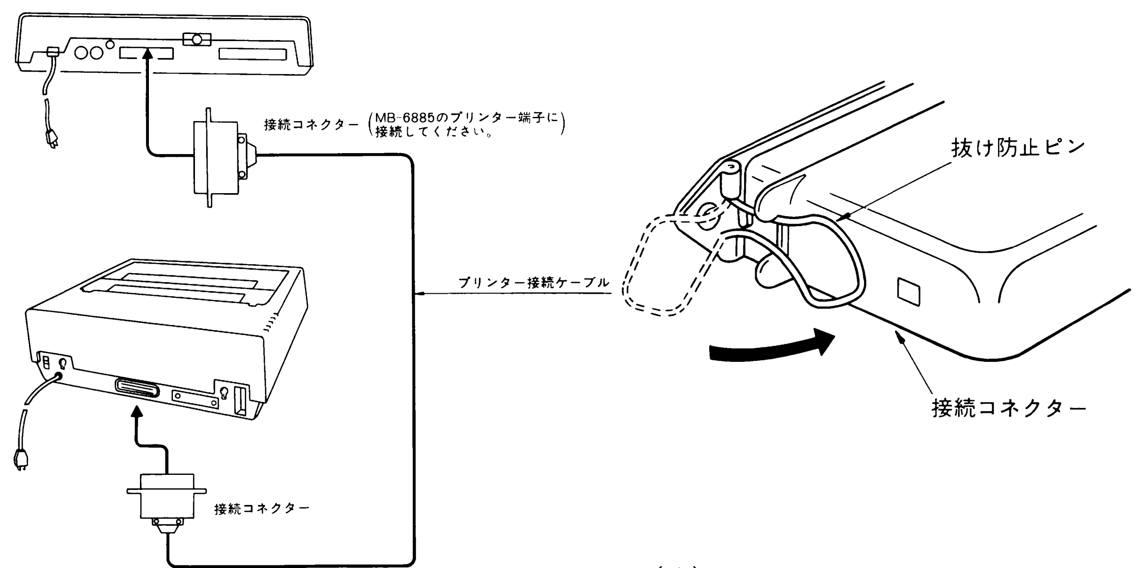
Y＼X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P		p	┼			ー	タ	ミ	時	
1			!	1	A	Q	a	q	└		。	ア	チ	ム	分	
2			"	2	B	R	b	r	┘		「	イ	ツ	メ	秒	
3			#	3	C	S	c	s	┐		」	ウ	テ	モ	年	
4			$	4	D	T	d	t	┌		、	エ	ト	ヤ	月	
5			%	5	E	U	e	u	─		・	オ	ナ	ユ	日	
6			&	6	F	V	f	v	│		ヲ	カ	ニ	ヨ		
7			'	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	8	H	X	h	x	┤		ィ	ク	ネ	リ		
9			)	9	I	Y	i	y	┬		ゥ	ケ	ノ	ル		
A			*	:	J	Z	j	z	├		ェ	コ	ハ	レ		
B			+	;	K	[	k		┴		ォ	サ	ヒ	ロ		
C			,	〈	L	¥	l				ャ	シ	フ	ワ		
D			−	=	M	]	m				ュ	ス	ヘ	ン		
E			.	〉	N	^	n				ョ	セ	ホ	゛		
F			/	?	O	_	o				ッ	ソ	マ	゜		

4．プリンターの使用例

詳しくは，ＭＰ－１０４１の取扱い説明書をご覧ください。また同梱包のフルグラフィックソフトテープは，ベーシックマスターレベル３専用のためベーシックマスタージュニアには使えません。

ａ．ベーシックマスタージュニアとの接続方法

接続のときは，ベーシックマスタージュニアおよびプリンターの電源を切って行ってください。またプリンター接続コネクターの端子には手を触れないでください。



（注）
接続ケーブルの接続コネクターを，プリンター本体およびＭＢ－６８８５本体に接続した後は，コネクターが抜けないよう，抜け防止ピンでコネクターを止めてください。

b．プログラム例

例121 ダイレクト実行でＬＦコードを転送する。

```
?#, CHR$($0A) RETURN
```

例122 ＳＩ側データコードとＳＯ側データコードの混在出力

```
10 FOR I=$91 TO $95
20 PRINT #, CHR$($0E);CHR$(I); ------------------------SO側での出力
30 NEXT I
40 FOR J=$91 TO $95
50 PRINT #, CHR$($0F);CHR$(J); ------------------------SI側での出力
60 NEXT J
70 PRINT #, CHR$($0A)
80 END
ｰｰ| |-‾ーあいうえお -------------------------------------------------------実行結果
```

例123 ＶＦＵ（バーチカルフォーマットユニット）に，タブ位置を設定するプログラムです。チャンネル１を５行，チャンネル２を10行，チャンネル３を15行に設定します。プログラムリストは１３２字モードで出力しています。

```
                    5スペース  10スペース      15スペース     2スペース
10 PRINT #,CHR$($14);"     1          2               3  123?";
20 PRINT #,CHR$($0B);"1"; --------------------------------チャンネルNo 1受信
30 LET A$="VFU VT CH"
40 FOR I=1 TO 2 ─┐
50 PRINT #,A$;"1" │------------------------------VFU  VT  CH1を2行印字
60 NEXT I ───────┘
70 PRINT #,CHR$($0B);"2"; --------------------------------チャンネルNo 2受信
80 FOR K=1 TO 9 ─┐
90 PRINT #,A$;"2" │------------------------------VFU  VT  CH2を9行印字
100 NEXT K ──────┘
110 PRINT #,CHR$($0B);"3"; -------------------------------チャンネルNo 3受信
120 PRINT #,A$;"3" ---------------------------------------VFU  VT  CH3を1行印字
130 END
```

実行結果

```
          ┐
          │
          │------------ 5改行
          │
          ┘
VFU VT CH1 }
VFU VT CH1 }------------------- 2行印字
          ┐
          │
          │------------ 8改行（10－2）
          │
          ┘
```

```
VFU VT CH2
VFU VT CH2
VFU VT CH2
VFU VT CH2
VFU VT CH2   ……………… 9行印字
VFU VT CH2
VFU VT CH2
VFU VT CH2
VFU VT CH2
                ……………… 6改行（15－9）
VFU VT CH3   ……………… 1行印字
```

例124 アンダーラインの印字

```
10 PRINT #,"ｱｲｳｴｵ";
20 PRINT #,CHR$($1B);"C";………アンダーライン印字開始コード
30 FOR I=$91 TO $95
40 PRINT #,CHR$($0F);CHR$(I);
50 NEXT I
60 PRINT #,CHR$($1B);"D";………アンダーライン印字終了コード
70 PRINT #,"ｱｲｳｴｵ"
80 END
ｱｲｳｴｵあいうえおｱｲｳｴｵ ………………実行結果
```

例125 ダイレクトスキップの使用例 〝ダイレクト　スキップ〟と印字して5行スキップし，ＥＮＤと印字します。

```
10 PRINT #,"ﾀﾞｲﾚｸﾄ ｽｷｯﾌﾟ"
20 PRINT #,CHR$($1B);CHR$($0B);"0";"5";
30 PRINT #,"END"
40 END
ﾀﾞｲﾚｸﾄ ｽｷｯﾌﾟ………………実行結果
                ……………… 5改行
END
```

例126 ページ長を½インチに設定

```
10 PRINT #,CHR$($1B);"F";"0";"1";  ------ページ長を½インチに設定
20 PRINT #,CHR$($1B);"5";  --------------------TOP OF FORM位置セット
30 FOR I=1 TO 2
40 FOR J=$91 TO $95
50 PRINT #,CHR$($0F);CHR$(J);
60 NEXT J
70 PRINT #,CHR$($0C);  -------------------------印字後次のページまで用紙を送る
80 NEXT I
90 END
```

```
あいうえお ----------------------------------------実行結果
            ------------------------------------½インチ幅に印字
あいうえお
```

例127 1/144インチ～20/144インチまでの改行

```
10 FOR I=1 TO 20
20 PRINT #,CHR$($1B);"%";"9";CHR$(I);  --------1/144インチ改行設定
30 PRINT #,"DOT IMPACT PRINTER"
40 NEXT I
50 PRINT #,CHR$($1B);"6";  --------------------1/6インチ改行設定
60 END
```

```
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
DOT IMPACT PRINTER
```

例128 15/144インチ改行と，SO側データコードを使用して罫線を印字します。プログラムリストは，132字モードでの出力です。

```
10 PRINT #,CHR$($0E)  ------------------------------SO側を指定
15 PRINT #,CHR$($1B);"%";"9";CHR$(15);  ------------15/144改行を指定
20 PRINT #,CHR$($98);CHR$($95);CHR$($95);CHR$($95);
30 PRINT #,CHR$($91);CHR$($95);CHR$($95);CHR$($95);
40 PRINT #,CHR$($91);CHR$($95);CHR$($95);CHR$($95);
50 PRINT #,CHR$($91);CHR$($95);CHR$($95);CHR$($95);CHR$($99)
60 PRINT #,CHR$($96);" A ";CHR$($96);" B ";CHR$($96);" C ";CHR$($96);" D ";
65 PRINT #,CHR$($96)
70 PRINT #,CHR$($93);CHR$($95);CHR$($95);CHR$($95);
80 PRINT #,CHR$($8F);CHR$($95);CHR$($95);CHR$($95);
90 PRINT #,CHR$($8F);CHR$($95);CHR$($95);CHR$($95);
100 PRINT #,CHR$($8F);CHR$($95);CHR$($95);CHR$($95);CHR$($92)
110 PRINT #,CHR$($96);"   ";CHR$($96);"   ";CHR$($96);"   ";CHR$($96);"   ";
115 PRINT #,CHR$($96)
120 PRINT #,CHR$($9A);CHR$($95);CHR$($95);CHR$($95);
130 PRINT #,CHR$($90);CHR$($95);CHR$($95);CHR$($95);
140 PRINT #,CHR$($90);CHR$($95);CHR$($95);CHR$($95);
150 PRINT #,CHR$($90);CHR$($95);CHR$($95);CHR$($95);CHR$($9B)
160 PRINT #,CHR$($1B);"6"  -------------------------1/6インチ改行に戻す
170 END
```

A	B	C	D

------------実行結果

例129 印字文字モードを変えた印字例です。実行結果は次頁をご覧ください。

```
 10 REM ** OPEN CLOSE 132/LINE **
 20 REM
 30 OPEN 1,7,$E009
 40 LET A=1
 50 GOSUB 420
 60 CLOSE 1
 70 REM
 80 REM ** OPEN CLOSE 80/LINE **
 90 REM
100 OPEN 2,7,$E000
110 LET A=2
120 GOSUB 420
130 CLOSE 2
140 REM
150 REM ** OPEN CLOSE 40/LINE **
160 REM
170 OPEN 3,7,$E012
180 LET A=3
190 GOSUB 420
200 CLOSE 3
210 REM
220 REM ** PRINT BY PRINT#, **
230 REM
240 REM ** 132/LINE MODE **
250 REM
260 PRINT #,CHR$($1B);"G";"ABCDEFG"
270 REM
280 REM ** 66/LINE MODE **
290 REM
300 PRINT #,CHR$($1B);"G";CHR$($1B);"U";"ABCDEFG"
310 REM
320 REM ** 80/LINE MODE **
330 REM
340 PRINT #,CHR$($1B);"R";"ABCDEFG"
350 REM
360 REM ** 40/LINE MODE **
370 REM
380 PRINT #,CHR$($1B);"R";CHR$($1B);"U";"ABCDEFG"
390 END
400 REM
410 REM ** PRINT **
420 PRINT #A,"ABCDEFG"
430 RETURN
```

実行結果

```
ABCDEFG ····························· 132字／行 ┐
ABCDEFG ····························· 80字／行  │ OPEN  CLOSEによる印字
ABCDEFG ····························· 40字／行  ┘

ABCDEFG ····························· 132字／行 ┐
ABCDEFG ····························· 66字／行  │
ABCDEFG ····························· 80字／行  │ PRINT＃，による印字
ABCDEFG ····························· 40字／行  ┘
```

プリンターの使い方について

1．ＯＰＥＮ／ＰＲＩＮＴ＃〈式〉，／ＣＬＯＳＥ

プリンターへのデータ出力は１０７ページに示すステートメントを用いて行います。〈式１〉は１～１５，〈式２〉は７～１５の範囲で他のＯＰＥＮステートメントで使用していない数値を選んで指定してください。また，ＰＲＩＮＴ＃〈式〉，ではＰＲＩＮＴ＃１，Ａ＄；Ｂ＄，Ｃ＄のように；および，の区切り記号は使用できません。

2．ＰＲＩＮＴ＃，について

このステートメントを実行するとプリンターの文字数／行モードを８０字／行に設定しますので，他の文字数／行モードにするときは１１２ページの **例１２９** のように文字数／行を指定し続けてデータを出力してください。なおＰＲＩＮＴ＃，では；および，の区切り記号は使用できますが，ＰＲＩＮＴ＃，の最後が；で終ったとき，データの印字ができないことがあります。このようなときにはＰＲＩＮＴ＃，〝ＡＢＣ〟；：ＰＲＩＮＴ＃，のように；で続いているデータの最後にＰＲＩＮＴ＃，を追加してください。

3．ＳＩ（シフトイン），ＳＯ（シフトアウト）モード設定について

ＯＰＥＮおよびＰＲＩＮＴ＃，ではステートメントを実行するとプリンターをＳＯ側に設定します。ＰＲＩＮＴ＃，によりＳＩ側を印字するには，１０９ページの **例１２２** のようにＳＩ側を指定し，続けてデータを出力します。ＯＰＥＮステートメントに続くＰＲＩＮＴ＃〈式〉，では，モードを変更しないかぎりそれまでに設定されたモードで印字します。

4．ＦＦ（フォームフィード機能）について

プリンターのページ長を指定してＦＦ（フォームフィード）を行うときには，ページ長は５インチ（標準改行量１／６インチで３０行）以内でご使用ください。５インチ以上を指定しますとＰＲＩＮＴＥＲ ＮＯＴ ＲＥＡＤＹが表示されますので，５インチ以上必要なときには５インチ以内のページ長を指定し，くり返し実行してご使用ください。

5．エラー表示について

プリンターが接続されていないとき，またはプリンターが動作可能な状態でないとき印字命令を実行すると約３秒後にＰＲＩＮＴＥＲ ＮＯＴ ＲＥＡＤＹが表示されます。ただし，このエラーはプリンターへのデータ出力中に急にプリンターが動作不可能状態となった場合は表示されませんので，プリンターの接続状態，ＳＥＬスイッチの状態を確認してください。

ヒント

ＰＲＩＮＴ ＃，ＣＨＲ＄（＄ＸＸ）で，プリンターに＄ＸＸのデータを送りますから，プリンターのデータコード表内の任意のデータを使用することができます。

例　市を印字する

ＰＲＩＮＴ ＃，ＣＨＲ＄（＄０Ｆ）；ＣＨＲ＄（＄83）；ＣＨＲ＄（＄０Ｄ）

3.13 高解像度グラフィックの使い方

ベーシックマスタージュニアは，縦１９２ドット，横２５６ドットで構成するグラフィック画面を，２ページ持つことができます。この機能を利用すると，47ページの表示コード表以外の文字や図形を自由に表示することができます。

1.グラフィックメモリーの領域定義

グラフィック機能を使用するということを，ベーシックマスタージュニアに宣言するものです。プログラムを他の機器からＬＯＡＤしたり，これから作る前に，お使いになるページ数により次の操作を行ってください。

ａ．グラフィックを１ページだけ使う場合

　　ＣＡＬＬ　＄Ｅ３７Ｅ RETURN とキー入力してください。

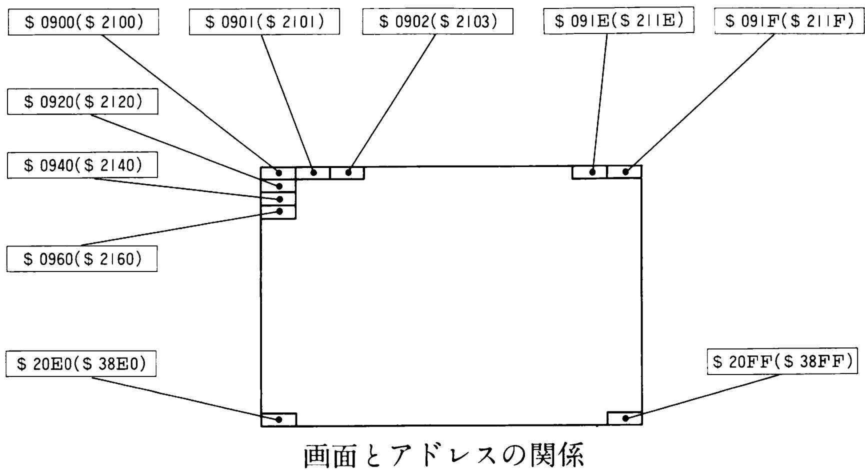
ｂ．グラフィックを２ページとも使う場合

　　ＣＡＬＬ　＄Ｅ３８３ RETURN とキー入力してください。

注1. グラフィック機能を使うとき，上記操作を忘れてプログラムを作りますと，プログラムを破壊しますのでご注意ください。

2.グラフィックメモリーの使用アドレス

図に示す様に画面をメモリー（ＲＡＭ）上に置きかえており，横方向に３２バイト，縦方向に１９２バイトで構成されています。したがって，画面左上端は＄０９００(2ページ目は＄２１００）番地，その隣が＄０９０１(2ページ目は＄２１０１）番地，画面右下端は＄２０ＦＦ(２ページ目は＄３８ＦＦ)番地となっています。それぞれのアドレスには，b_7～b_0までの８ビットのデータビットを持っており，このデータビットと前述のドットが対応するわけです。データビットに１を書き込めば画面上のドットが光り，０を書き込むと消えます。



画面とアドレスの関係

各アドレスのデータビット構成

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

ヒント

横方向に８ドットずれた位置を求めるには　今のアドレス±１　例 ＄2102－１＝＄2101

縦方向に　１列　ずれた位置を求めるには　今のアドレス±＄20　例 ＄0982＋＄20＝＄09A2

３.データビットと画面の状態

画面左上端部に，ドットを横方向に光らせる(以下白とする)部分と，消す（以下黒とする）部分を交互に出すパターンを作るには，次の様にして行います。グラフィックの１ページを使うとすると，画面左上端部のアドレスは，＄０９００番地です。このアドレスにb_7～b_0の８ビットのデータビットがありますので，左から白黒白黒白黒白黒と表示するには，b_7が１，b_6が０，b_5が１，b_4が０，b_3が１，b_2が０，b_1が１，b_0が０とする必要があります。b_7～b_0のデータビットは，b_7～b_4，を上位４ビット，b_3～b_0を下位４ビットと呼び，４ビット毎に１６進数で表現します。

図にあるように＄０９００番地に＄ＡＡを書き込むと１０１０１０１０すなわち，白黒白黒白黒白黒のパターンが横方向に表示できます。

アドレス　＄０９００

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
データビット	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
データ	1	0	1	0	1	0	1	0
画面上のパターン								
	上位ビット				下位ビット			

上位４ビットデータ１０１０は右表よりＡ
下位４ビットデータ１０１０は右表よりＡ

＄０９００に＄ＡＡというデータを書き込む

```
POKE　＄900，＄AA
```

１０進数と１６進数について７５ページのヒントも参照してください。

データビット

16進表示	b_7 / b_3	b_6 / b_2	b_5 / b_1	b_4 / b_0	10進表示
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
A	1	0	1	0	10
B	1	0	1	1	11
C	1	1	0	0	12
D	1	1	0	1	13
E	1	1	1	0	14
F	1	1	1	1	15

４.表示モードの切り替え

表示モードは，グラフィック１，グラフィック２，テキスト（通常の文字・図形モード）の３種類があります。目的に合せて，次の様に操作してください。ダイレクト実行でも可能です。

ａ．グラフィック１ページ目の選択

POKE SEFE0，＄C0

又は，POKE SEFE0，192

ｂ．グラフィック２ページ目の選択

POKE＄EFE0，＄CC

又は，POKE＄EFE0，204

c．テキストモードの選択

POKE$EFE0，0

5.グラフィックメモリーのクリア

グラフィックメモリーに書き込んだデータをすべて消したい場合に使います。テキストモードでのCLEARステートメントと同じ働きをします。

a．グラフィック1ページ目のクリア

CALL$E38D

b．グラフィック2ページ目のクリア

CALL$E39C

例130 縦8ドット，横8ドットの8×8ドットの構成で，♠パターンを画面左上端に作ります。

```
     CALL  $E37E [RETURN] ……………… グラフィックページを1ページ確保
  10 CALL  $E38D ……………………… グラフィック1ページ目をクリア
  20 POKE  $EFE0，$C0 …………… グラフィック1ページ目を選択
  50 POKE  $900，$00 ……………… POKE  $900，0でも可
  60 POKE  $920，$18
  70 POKE  $940，$3C
  80 POKE  $960，$7E
  90 POKE  $980，$FF
 100 POKE  $9A0，$24
 110 POKE  $9C0，$42
 120 POKE  $9E0，$81
 160 FOR   K=1 TO 2000 ─┐
 170 NEXT  K ───────────┘ この期間♠を表示
 180 POKE  $EFE0，0 ……………… テキストモードに戻す
 190 END
```

画面上のパターン

アドレス	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	データ
$0900	0	0	0	0	0	0	0	0	$00
$0920	0	0	0	1	1	0	0	0	$18
$0940	0	0	1	1	1	1	0	0	$3C
$0960	0	1	1	1	1	1	1	0	$7E
$0980	1	1	1	1	1	1	1	1	$FF
$09A0	0	0	1	0	0	1	0	0	$24
$09C0	0	1	0	0	0	0	1	0	$42
$09E0	1	0	0	0	0	0	0	1	$81

（データビット）

プログラム中この0は省略可

例131　**例130**　の５０行目から１２０行目を次の様に書き換えて３０行目を挿入すれば同じです。

```
 10 CALL $E38D
 20 POKE $EFE0,$C0
 30 A=$900 ------------------------ 表示先頭アドレスを変数に代入
 50 POKE A,0
 60 POKE A+$20,$18
 70 POKE A+$40,$3C
 80 POKE A+$60,$7E
 90 POKE A+$80,$FF
100 POKE A+$A0,$24
110 POKE A+$C0,$42
120 POKE A+$E0,$81
160 FOR K=1 TO 2000
170 NEXT K
180 POKE $EFE0,0
190 END
```

前のプログラムを消さずに次のプログラムをキー入力し，実行してみてください。♠が画面を移動します。移動速度は１３０行目を変えることにより変化します。

```
 30 FOR A=$900 TO $201F STEP $1F
 40 CALL $E38D
130 FOR J=1 TO 100 ------------- この値を変えると速度が変る
140 NEXT J
150 NEXT A
```

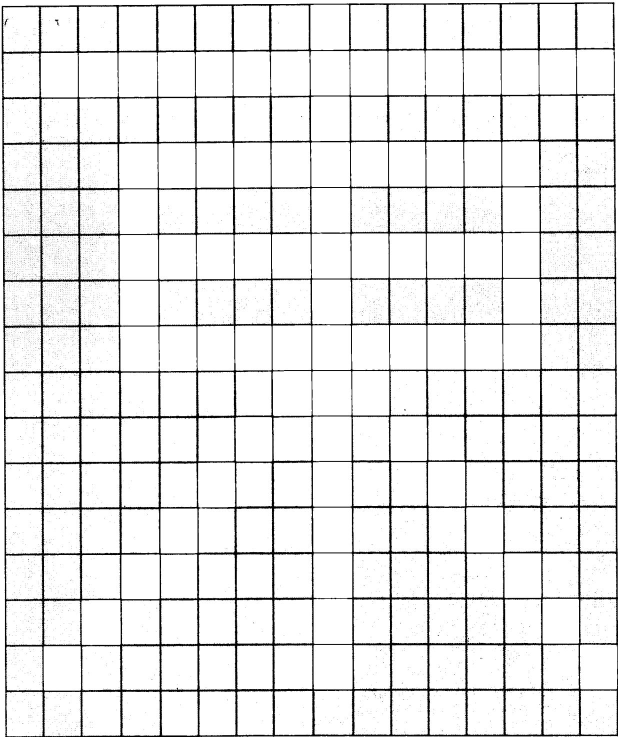
３０行目を次の様にして実行してみてください。

```
 30 FOR I=$900 TO $201F STEP $100
```

例132 縦16ドット，横16ドットの16×16ドット構成で，東という漢字を作ります。横に16ドットを取るためには，1アドレス8ビットですから，横に2アドレスを使い，縦方向には，1段1アドレスですから，16アドレスを使います。このプログラムは **例134** まで使用しますので消さない様にしてください。

```
   CALL $E383 [RETURN]---------------グラフィックページを2ページ確保
10 CALL $E38D------------------------グラフィック1ページ目をクリア
20 CALL $E39C------------------------グラフィック2ページ目をクリア
30 POKE $EFE0,$C0--------------------グラフィック1ページ目を選択
40 LET A=$900------------------------漢字表示の先頭アドレスを変数に代入
50 POKE A,0
60 POKE A+1,$80
70 POKE A+$20,$7F
80 POKE A+$21,$FF
90 POKE A+$40,0
100 POKE A+$41,$80
110 POKE A+$60,$1F
120 POKE A+$61,$FC
130 POKE A+$80,$10
140 POKE A+$81,$84
150 POKE A+$A0,$1F
160 POKE A+$A1,$FC
170 POKE A+$C0,$10
180 POKE A+$C1,$84
190 POKE A+$E0,$1F
200 POKE A+$E1,$FC
210 POKE A+$100,$01
220 POKE A+$101,$C0
230 POKE A+$120,$02
240 POKE A+$121,$A0
250 POKE A+$140,$04
260 POKE A+$141,$90
270 POKE A+$160,$08
280 POKE A+$161,$88
290 POKE A+$180,$10
300 POKE A+$181,$84
310 POKE A+$1A0,$20
320 POKE A+$1A1,$82
330 POKE A+$1C0,$40
340 POKE A+$1C1,$81
350 POKE A+$1E0,0
360 POKE A+$1E1,0
540 FOR I=1 TO 5000--┐
550 NEXT I-----------┘---------------この期間 東 を表示
560 POKE $EFE0,0---------------------テキストモードへ戻す
570 END
```

画面上のパターン



		アドレス1 データビット								アドレス2 データビット									
アドレス1	データ	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	アドレス2	データ
$0900	$00	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	$0901	$80
$0920	$7F	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	$0921	$FF
$0940	$00	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	$0941	$80
$0960	$1F	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	$0961	$FC
$0980	$10	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	$0981	$84
$09A0	$1F	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	$09A1	$FC
$09C0	$10	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	$09C1	$84
$09E0	$1F	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	$09E1	$FC
$0A00	$01	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	$0A01	$C0
$0A20	$02	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	$0A21	$A0
$0A40	$04	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	$0A41	$90
$0A60	$08	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	$0A61	$88
$0A80	$10	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	$0A81	$84
$0AA0	$20	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	$0AA1	$82
$0AC0	$40	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	$0AC1	$81
$0AE0	$00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$0AE1	$00

例133 例132 をそのままに，次のプログラムを追加して実行してみてください。

```
370 LET B=$B00
380 POKE B,0,$80
390 POKE B+$20,$7F,$FF
400 POKE B+$40,0,0
410 POKE B+$60,$1F,$FC
420 POKE B+$80,$10,4
430 POKE B+$A0,$10,4
440 POKE B+$C0,$10,4
450 POKE B+$E0,$1F,$FC
460 POKE B+$100,0,$80
470 POKE B+$120,0,$80
480 POKE B+$140,4,$90
490 POKE B+$160,8,$88
500 POKE B+$180,$10,$84
510 POKE B+$1A0,$21,$82
520 POKE B+$1C0,0,$80
530 POKE B+$1E0,0,0
```

例134 東京という漢字を，グラフィック２ページ目を使って横書きにしてみます。

３０行目，４０行目，３７０行目を修正し，５６０行目以降を追加して，３０行目から実行してください。

```
 30 POKE $EFE0,$CC
 40 A=$2300
370 B=$2302
560 FOR J=1 TO 5
570 POKE $EFE0,$C0
580 FOR K=1 TO 1000
590 NEXT K
600 POKE $EFE0,$CC
610 FOR L=1 TO 1000
620 NEXT L
630 NEXT J
640 POKE $EFE0,0
650 END
```

Ｒ３０ RETURN とキー入力し実行してください。

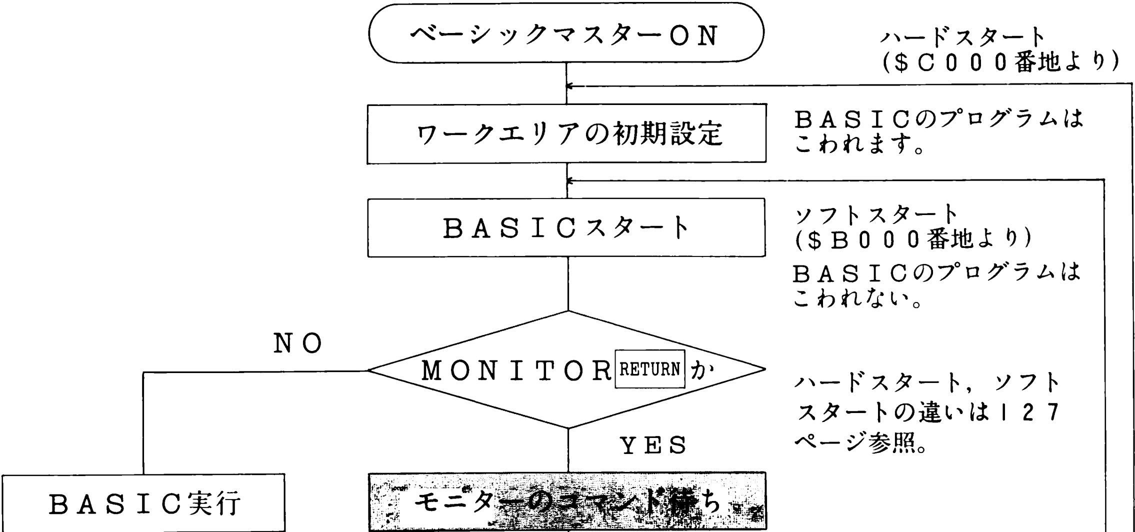
3.14 エラーコード

No.	エラー表示	エラー内容概要
1	SYNTAX ERROR (構文規則違反)	●ステートメントの区切り記号が不適当である。 ●キーワードの綴りに誤りがある。 ●文字列が〝 〞ではさまれていない。 ●コマンドの最後が RETURN で終っていない。 ●ＧＯＴＯ，ＧＯＳＵＢ文の分岐先行番号 RETURN で終っていない。 ●ON～GOTO(GOSUB)文で分岐先行番号の区切り記号が不適当である。 ●ＬＩＳＴで存在しない行を表示しようとした。 ●ＯＮの次に〈式〉がない。また，ＧＯＴＯ，ＧＯＳＵＢ以降がない。 ●ＩＦ文でＴＨＥＮ以降がない。 ●ＦＯＲ文でＴＯがない。 ●ＤＥＦ文でＦＮの次が英字でない。また仮引数が数値変数でない。 ●〈変数〉＝で変数の後に＝以外の文字がある。 ●〈式〉の記述が誤っている。 ●ＭＵＳＩＣ文で音楽記号以外がある。表現できない高さの音を指定した。 ●その他構文規則エラー。
2	LINE NUMBER ERROR	●ＧＯＴＯ，ＧＯＳＵＢで指定された分岐先番号がみつからない。 ●ＯＮ～ＧＯＴＯ（ＧＯＳＵＢ）で指定された分岐先番号がみつからない。 ●行番号が１～３２７６７の範囲にない。
3	OVERFLOW ERROR	●入力した数値の絶対値が大きすぎる。 ●整数(－３２７６７～＋３２７６７)の範囲を超えている。 ●整数(０～６５５３５)を超えている。 ●計算途中あるいは計算結果が扱える値の範囲を超えている。
4	MEMORY FULL ERROR	●プログラムサイズが大きすぎる。 ●変数エリアがたりない。
5	ERROR 1	●変数名が多すぎる。約128個以下であること。
6	ERROR 2	●ステートメントが長すぎる。（７９字を超えた） ●文字列が長すぎてバッファーが一杯になった。（７９字を超えた）
7	ERROR 3	●ファイル名がおかしい。（６文字を超えている）
8	ERROR 4	●零で割ろうとした。
9	ERROR 5	●関数の引数（の値）が許されない値である。
10	ERROR 6	●ユーザ指定デバイスのドライブルーチンがない。 ●OPENされていないのにPRINT＃，INPUT＃を実行しようとした。 ●デバイスが接続されていない。 ●デバイスがOPENされているのに，再度OPENしようとした。

No.	エラー表示	エラー内容概要
11	ERROR 7	●単純変数として使われているのに配列として参照した。 ●配列宣言されているのに（ ）なしで参照した。
12	ERROR 8	●ＦＯＲ文なしでＮＥＸＴ文を実行した。入れ子構造になっていない。 ●ＦＯＲ文とＮＥＸＴ文の制御変数が一致しない。
13	ERROR 9	●ＧＯＳＵＢ文で呼ばれていないのにＲＥＴＵＲＮ文を実行した。
14	ERROR 10	●文字式の中に文字因子以外が書かれている。
15	ERROR 11	●ＦＯＲ—ＮＥＸＴのネストが深すぎる。（最大15重） ●〈式〉が複雑すぎる。
16	ERROR 12	●サブルーチンのネストが深すぎる。（最大30重）
17	ERROR 13	●変数が未定義である。
18	ERROR 14	●配列の２重定義をした。（既に配列として宣言されている）
19	ERROR 15	●配列宣言でサイズ指定が大き過ぎる。（数値：1～8500 文字：1～255） ●配列宣言でサイズ指定が0または負である。
20	ERROR 16	●配列が宣言されていない。
21	ERROR 17	●配列参照において，宣言されているサイズを超えた指定をした。
22	ERROR 18	●文字変数を演算しようとした。
23	ERROR 19	●未定義関数を使用している。 ●〈式〉の中で使えない文字取扱い関数を使用している。
24	ERROR 20	●未定義のユーザ定義関数を参照した。 ●ユーザ定義関数が異常なループを形成している。
25	ERROR 21	●ＤＡＴＡ文がない。ＤＡＴＡの数がたりない。
26	ERROR 22	●ＤＡＴＡ文でＤＡＴＡの区切りがおかしい。
27	ERROR 23	●数値以外のデータを数値として扱おうとした。
28	ERROR 24	●ＩＮＰＵＴ，ＧＯＳＵＢ文をダイレクト実行しようとした。
29	ERROR 25	●コマンドをプログラム中で実行しようとした。
30	PRINTER NOT READY	●プリンターが接続されていない。 ●プリンターが動作可能状態でない。

4.モニターの使い方(V1.2)

BASICのコマンドで　>MONITOR（省略形MON）RETURN とキー入力することにより，ベーシックマスタージュニアはモニターに制御が移り，モニターのコマンド(命令) 待ちになります。



コマンド名（働き）	概　　　　要
B（BREAK）	ブレークポイントの設定，解除。
D（DISPLAY）	指定番地以下128バイトの表示。
E（ESCAPE）	BASICにジャンプ。
F（FILL）	指定されたメモリーブロックに定数を記録。
G（GO）	指定した番地からプログラムを実行。
L（LOAD）	6文字以内で指定したファイル名のプログラムをテープから入力。
M（MEMORY）	指定された番地の内容の表示，および変更。
P（PUNCH）	6文字以内のファイル名で指定したプログラムをテープに出力。
R（REGISTOR）	MPUレジスターの内容表示，および変更。
S（STEP）	プログラムカウンターの示すアドレスからの1命令実行。
T（TRANSFER）	指定したメモリーブロックを他のアドレス領域に転送。
V（VERIFY）	SAVEしたテープとメモリ内容との比較。

モニターでは，BASICの場合と RETURN スペース キーの働きが異なります。一般的にモニターでは，スペースキーが現在実行中のコマンドの進行，処理を行ない，RETURN キーは現在実行中のコマンドを終了し，次のコマンドのためのコマンド待ちに戻ります。

また，RESETはEコマンドと同じように，BASICの初期状態にジャンプする働きをします。DEL キーもBASICの場合と多小異なります。一文字削除でなくキー入力した文字あるいは数字を削除します。

4.1 メモリーマップ

ベーシックマスタージュニアのメモリー構成と，ユーザーRAMエリアは下図のようになっています。64KB実装でのRAMモード切替えは，オプションのRAMキット取扱説明書をご覧ください。

アドレス	エリア
\$0000〜\$00FF	システムワークエリア
\$0100〜\$03FF	テキストモードエリア
\$0400〜\$09FF	ベーシックワークエリア
\$0A00〜\$21FF	グラフィック1ページエリア
\$2200〜\$39FF	グラフィック2ページエリア
\$3A00〜RAMEND	ユーザーRAMエリア
RAMEND〜\$AFFF	
\$B000〜\$DFFF	BASIC ROM
\$E000〜\$E7FF	プリンター ROM
\$E800〜\$EDFF	外部 I/O
\$EE00〜\$EFFF	システム I/O
\$F000〜\$FFFF	モニター ROM

注1.標準実装RAMは16KBです。

注2.テキストモードは，キャラクタゼネレータにより文字，数字，図形を表示するモードです。

注3.1KBは1,024バイトを表わします。

注4.下表のユーザーRAMエリアは RAMTOP〜RAMEND 表示です。

注5.ユーザーRAMエリアは，ユーザーが実際に使える領域で，システムワークエリア等は含んでおりません。

モード		ユーザーRAMエリア(バイト)	
		RAM16KB実装	RAM64KB実装
BASIC動作時	テキストモード時(標準状態)	\$0A00〜\$3FFF (13,824)	\$0A00〜\$AFFF (42,496)
	グラフィック1ページ使用時	\$2200〜\$3FFF (7,680)	\$2200〜\$AFFF (36,352)
	グラフィック2ページ使用時	\$3A00〜\$3FFF (1,536)	\$3A00〜\$AFFF (30,208)
モニター動作時	BASIC ROMをRAM化		\$0400〜\$DFFF (56,320)
	プリンターROM 外部I/OをRAM化		\$0400〜\$EDFF (59,904)
オール RAM化			\$0000〜\$EDFF \$F000〜\$FFFF (65,024)

4.2 コマンド仕様

1. B(BREAK)コマンド

1. 機能
   a. 設定中のブレークポイントの表示，設定，変更を行ないます。
   b. ブレークポイントは一度に5ヶ所まで設定できます。

2. 操作
   a. モニターのコマンド待ちの状態で[B]をキー入力することにより，現在設定されているブレークポイントが表示されます。
   b. 新しいブレークポイントを設定する場合，アドレスをキー入力した後，スペースを入力してください。（スペースキーをおす。）
   c. すでに設定されているブレークポイントを解除する場合，解除したいアドレスをキー入力した後，スペースを入力してください。
   d. 全てのブレークポイントを解除する場合，アドレス0番地をキー入力した後，スペースを入力してください。
   e. コマンドを終了するときは[RETURN]をキー入力してください。

＊ブレークポイントとは，機械語のプログラムにおけるＢＡＳＩＣのＳＴＯＰステートメントのような働きをするコマンドで，プログラムの作成中またはデバックのとき適時ブレークポイントを設け，レジスターの内容などをチェックするためにあります。

使用例

1. ＄１Ａ００と＄１ＢＡＦ番地をブレークポイントに設定。

```
**   MONITOR        V1.2
* INPUT COMMAND     B

* BREAK POINT
 - 1A00
* BREAK POINT
 - 1A00
 - 1BAF
* BREAK POINT
 - 1A00
 - 1BAF
 - *
```

2. ブレークポイントを解除。

```
**   MONITOR        V1.2
* INPUT COMMAND     B

* BREAK POINT
 - 1A00
 - 1BAF
 - 0000
* BREAK POINT
 - *
```

2. D(DISPLAY)コマンド

1. 機能
   a. 指定番地以下のメモリーの内容を１画面(１２８バイト)に表示します。

2. 操作
   a. モニターのコマンド待ちの状態で D をキー入力するとディスプレイコマンドが起動し，ＤＩＳＰＬＡＹ　ＦＲＯＭ　××××　－　＊と表示され，アドレス入力要求のカーソルが点めつします。
   b. 表示したいアドレスをキー入力し，つぎにスペースを入力すると，キー入力したアドレスから１行あたり８バイトで計１６行，１２８バイトのブロックを表示し，最後に　ＤＩＳＰＬＡＹ　ＦＲＯＭ　○○○○　－＊と，次のブロックのアドレスが表示されます。
   c. 表示されたアドレスからつづいてディスプレイするときはスペースを入力します。ディスプレイするアドレスを変更するときは，アドレスをキー入力しつぎにスペースを入力してください。
   d. コマンドを終了するときは RETURN をキー入力してください。

使用例

あらかじめＦコマンドにより，＄４００番地から＄３ＦＦＦ番地にＦＦを書き，次にＥコマンドでＢＡＳＩＣにジャンプし，４８ページの **例51** を作ってＡからＪまでを３２個ずつ書き込んだときの配列エリアの様子です。

```
* INPUT COMMAND    D

 DISPLAY FROM 03E2   - 3E80

3E80 FF FF FF FF FF FF FF FF
3E88 FF FF FF FF FF FF FF FF
3E90 FF FF FF FF FF FF FF FF
3E98 FF FF FF FF FF FF FF FF
3EA0 FF FF FF FF FF FF FF FF
3EA8 FF FF FF FF FF FF 17 00
3EB0 00 0B 00 00 0A 20 41 41
3EB8 41 41 41 41 41 41 41 41
3EC0 41 41 41 41 41 41 41 41
3EC8 41 41 41 41 41 41 41 41
3ED0 41 41 41 41 41 41 20 42
3ED8 42 42 42 42 42 42 42 42
3EE0 42 42 42 42 42 42 42 42
3EE8 42 42 42 42 42 42 42 42
3EF0 42 42 42 42 42 42 42 20
3EF8 43 43 43 43 43 43 43 43
 DISPLAY FROM 3F00   - *
```

右図は，＄３ＦＦＦ※番地までの配列が格納されている様子です。
(※ＲＡＭ16Ｋバイト実装の場合)

```
3F68 46 46 46 46 46 46 46 46
3F70 46 46 46 46 46 46 46 46
3F78 46 46 46 20 47 47 47 47

 DISPLAY FROM 3F80   - 3F80

3F80 47 47 47 47 47 47 47 47
3F88 47 47 47 47 47 47 47 47
3F90 47 47 47 47 47 47 47 47
3F98 47 47 47 47 20 48 48 48
3FA0 48 48 48 48 48 48 48 48
3FA8 48 48 48 48 48 48 48 48
3FB0 48 48 48 48 48 48 48 48
3FB8 48 48 48 48 48 20 49 49
3FC0 49 49 49 49 49 49 49 49
3FC8 49 49 49 49 49 49 49 49
3FD0 49 49 49 49 49 49 49 49
3FD8 49 49 49 49 49 49 20 4A
3FE0 4A 4A 4A 4A 4A 4A 4A 4A
3FE8 4A 4A 4A 4A 4A 4A 4A 4A
3FF0 4A 4A 4A 4A 4A 4A 4A 4A
3FF8 4A 4A 4A 4A 4A 4A 4A FF
 DISPLAY FROM 4000   - :
```

3. E(ESCAPE)コマンド

1．機能
　a．ＢＡＳＩＣにもどります。

2．操作
　a．モニターのコマンド待ちの状態で E をキー入力すると，ＢＡＳＩＣに制御が移ります。この状態はＢＡＳＩＣモードの初期状態，電源ＯＮ時および，ＲＥＳＥＴによって得られる状態と同じです。

＊Ｅコマンドは，ＲＥＳＥＴ機能と等価的に同じ動作をします。従って，ＢＡＳＩＣのプログラムはＥコマンドを実行した段階で消去されます。ＢＡＳＩＣからモニターに制御を移す前にＢＡＳＩＣのプログラムをテープにＳＡＶＥするようにしてください。

4. G(GO)コマンド

1．機能
　a．指定した番地からプログラムを実行します。

2．操作
　a モニターのコマンド待ちの状態で， G をキー入力すると，Ｇコマンドが起動し，プログラムカウンターの示すアドレスとプログラムカウンター（スタートアドレス）の変更入力待ちを示す表示が表われます。

　　ＳＴＡＲＴ　ＡＤＤＲ　：　××××　－　＊

　b．表示されたアドレスより実行するときは，スペースを入力し，スタートアドレスを変更するときは，アドレスをキー入力した後，スペースを入力してください。

＊Ｇコマンドで実行したプログラム途中で停止する場合は以下のときです。
　1．ブレークポイントが設定されている場合，そのアドレスの一つ前まで実行し，レジスターの内容を表示して停止する。
　2．ＢＲＥＡＫキーを押したとき。
　3．ＳＷＩ命令を実行したとき。レジスターの内容を表示して停止する。
＊Ｇコマンドで＄Ｂ０００番地から実行すると，それ以前に入っているＢＡＳＩＣのプログラムをこわさずにＢＡＳＩＣに戻ることができます。このように戻る方法をソフトスタートと呼んでいます。それに対して，Ｇコマンドを＄Ｃ０００番地から実行すると，ＥコマンドやＲＥＳＥＴで戻ったときと同じ動作をし，ＢＡＳＩＣのプログラムはこわれます。この方法をハードスタートと呼んでいます。

5. F (FILL) コマンド

1．機能

a．指定されたメモリーブロックに定数を記録します。

2．操作

a．モニターのコマンド待ちの状態で [F] をキー入力するとFコマンドが起動し，記録する定数の入力待ちを表す　DATA　－　＊　が表示されます。

b．データをキー入力しつぎにスペースを入力すると，記録するメモリーブロックの先頭番地の入力待ちを示す　FROM　××××　－　＊　が表示されます。

c．先頭番地をキー入力し，つぎにスペースを入力すると，記録するメモリーブロックの終了番地の入力待ちを示す　TO　△△△△　－　＊　が表示されます。

d．終了番地をキー入力し，つぎにスペースを入力すると，Fコマンドを実行し，モニターのコマンド待ちになります。

e．正しく記録できなかった場合は，そのアドレスと内容を表示し，モニターのコマンド待ちになります。

注．b，c，項の××××番地，△△△△番地は，はじめは，初期設定されているROMの番地が表示されますが，無視してください。
　ROMの番地に記録しようとすると，e 項が適用されます。

使用例

1．データ“FF”を書き込みます。

```
**   MONITOR        V1.2
* INPUT COMMAND     F
DATA - FF*
```

2．FILL（満たす）アドレスの範囲を＄400番地から＄3FFF番地までします。

```
**   MONITOR        V1.2
* INPUT COMMAND     F
DATA - FF
FROM 0142   - 0400
 TO  0144   - 3FFF
* INPUT COMMAND     *
```

3．次にDコマンドで，FFが満たされた様子を見てください。

[D] [スペース] [スペース]…でOKです。

6. M(MEMORY)コマンド

Ⅰ. 機能

a. メモリーの指定された番地の内容を表示します。

b. メモリーの内容の変更，書き込みをします。

2. 操作

a. モニターのコマンド待ちの状態で[M]をキー入力するとMコマンドが起動しアドレスの入力待ちになります。

INPUT ADDR & DATA

－ ＊

b. アドレスをキー入力し，スペースを入力すると，メモリーの内容が表示され，データの入力待ちとなります。

－ ○×△□ ： FF － ＊

c. データを変更するときは，データをキー入力し，スペースを入力します。変更しないときは，スペースだけを入力してください。

d. c項の結果，a項と同じようにアドレスの入力待ちとなります。b項で表示されたアドレス（○×△□）の次のアドレス（○×△□＋Ⅰ）番地を表示するときは，スペースを入力します。新たなアドレスを表示したいときは，アドレスをキー入力した後，スペースを入力してください。

e. 以下，c項，d項を必要にくり返してください。

f. 以上のプロセスのとき[RETURN]キーを押すと，Mコマンドを終了し，モニターのコマンド待ちに戻ります。

注，メモリーの実装されていないアドレスエリア，ROMのアドレスエリアの内容を変更しようとした場合，データは変更されず？を表示し，警告音を発します。

＊Mコマンドにより，機械語プログラムを書き込む場合，ベーシックマスタージュニアで使用しているシステムワークエリア，BASICのプログラムエリア等を配慮してアドレスエリアを確保してください。詳しくはⅠ24ページのメモリーマップ，Ⅰ35ページ，機械語についての項をご参照ください。

＊＊Mコマンドによりシステムワークエリア，システムI／0エリア，BASICワークエリア内にデータを書き込みますと正しい動作をしないことがありますのでご注意ください。

使用例

Ⅰ. プログラムを書き込んでいるところです。

```
**   MONITOR        V1.2
* INPUT COMMAND    M
INPUT ADDR & DATA
 - 1A00  :  FF   -  BD
 - 1A01  :  FF   -  F0
 - 1A02  :  FF   -  0F
 - 1A03  :  FF   -  BD
 - 1A04  :  FF   -  00
 - 1A05  :  FF   -  2B
 - 1A06  :  FF   -  20
 - 1A07  :  FF   -  F8
 - *
```

スペースキーによりカーソルがここに出る　この結果、ⅠA07の変更が成立

7. R(REGISTOR)コマンド

1. 機能

  a. レジスター内容の表示，および変更をします。

2. 操作

  a. モニターのコマンド待ちの状態で[R]をキー入力するとRコマンドが起動し，レジスターの内容がＳＰ（スタックポインター），ＣＣ（コンディションコード），Ｂ（ＡＣＣ　Ｂ），Ａ（ＡＣＣ　Ａ），ＩＸ（インデックスレジスター），ＰＣ（プログラムカウンター）の順に表示されます。

  b. レジスターの内容が表示された後，ＳＰの表示の下に＊がブリンクし，変更入力待ちとなります。

  c. レジスターのデータを変更するときは，データをキー入力しスペースを入力します。また変更しないときはスペースだけを入力すると，入力待ちの＊が次のレジスターの表示の下に移ります。

  d. 以下，CC，B，A，IX，PCについてもc項と同様の操作を行ってください。

  e. ＰＣの設定が終了してスペースキーを入力した場合，または，途中で[RETURN]キーを押した場合，モニターのコマンド待ちに戻ります。

  f. スタックポインターのデータを実装ＲＡＭの最大アドレスよりも大きい値に変更しようとすると，？を表示し警告音を発します。

8. S(STEP)コマンド

1. 機能

  a. 割込みにより，プログラムをプログラムカウンターの示すアドレスから1命令ずつトレースします。

  b. ブレークポイントが設定されていても，1命令トレースを実行します。

2. 操作

  a. Ｓコマンドに先だち，ＲコマンドによりＰＣ（プログラムカウンター）の内容を，Ｓコマンドを実行するプログラムの先頭番地に変更してください。※

  b. モニターのコマンド待ちの状態で[S]をキー入力すると，Ｓコマンドが起動し，下図のように表示されます。

  c. つぎに，スペースを入力すると，b項でＰＣの示すアドレスのステップが表示されます。

  d. [RETURN]キーをおすと，モニターのコマンド待ちに戻ります。

```
**    MONITOR          V1.2
* INPUT COMMAND     R
    SP   CC   B   A   IX     PC
   3FFF  CB  F4  32  F20D  F1D3
 - 3FFF  CB  F4  32  F20D  1A00
* INPUT COMMAND     S

* STEP    1A00   :  BD
    SP   CC   B   A   IX     PC
   3FFD  CB  F4  32  F20D  F00F  *
```

※ＰＣの変更は，ＰＣ設定後，スペースの入力によって実行されます。[RETURN]不可。

9. T (TRANSFER) コマンド

1．機能

a．指定されたメモリーブロックの内容を他のメモリーブロックへ転送します。

2．操作

a．モニターのコマンド待ちの状態で T をキー入力するとTコマンドが起動し，転送するメモリーブロックの先頭番地のアドレス入力待ちとなります。

MASTER BLOCK ADDR
FROM ×××× － *

b．アドレスをキー入力して、スペースを入力すると転送するメモリーブロックの最終番地のアドレス入力待ちとなります。

TO △△△△ － *

c．アドレスをキー入力してスペースを入力すると，転送先のメモリーブロックの先頭番地のアドレス入力待ちとなります。

TRANSFER BLOCK ADDR
FROM □□□□ － *

d．アドレスをキー入力してスペースを入力すると，転送を開始し転送が終了するとモニターのコマンド待ちに戻ります。

e．正しいデーター転送が行なわれなかった場合，一致しない転送先のアドレスと，その内容が表示され，モニターのコマンド待ちに戻ります。

*正しく転送されない場合とは，実装されていないメモリーのアドレス領域に転送しようとした場合とか，ROM領域に転送しようとした場合などです。

*＄0番地から＄FF番地は，システムのワークエリアとして使用しています。このエリアに定数を転送するとプログラムが暴走します。ご注意ください。

4.3 プログラムの記録・再生コマンド

プログラムの記録，再生のコマンドはBASICのSAVE，VERIFY，LOADコマンドとほぼ同じような働きをし，モニターではそれぞれ，P，V，Lコマンドがその役割りを果します。モニターの他のコマンドにおいては，スペースキーがコマンドを実行する場合の基本的な役割りを果し，RETURN キーはコマンドを終了し，コマンド待ちに復帰するためのRESETキー的な役割りを持っていました。

しかし，P，V，Lコマンドにおいては RETURN キーの働きが異なりますのでご注意ください。

実際のプログラムの記録，再生のハード的な仕様はBASICと全く同じです。異なるのは，BASICの場合，ファイル名のあとに，Sが表示されていましたが，モニターの場合，B（バイナリープログラムを意味する。）が付加されることです。

エラーが発生したとき等の処置についてはBASICの項，97ページ以降をご参照ください。

1. P (PUNCH) コマンド

1. 機能

 a. 6文字以内のファイル名で定義されたプログラムをテープに記録します。

2. 操作

 a. モニターのコマンド待ちの状態で [P] をキー入力すると，Pコマンドが起動し，ファイル名の入力待ちとなります。 このとき [RETURN] キーを押してもコマンド待ちに戻りません。

  PROGRAM NAME : *

 b. プログラムのファイル名※をキー入力し，スペース（または [RETURN]）を入力すると，記録するメモリーの先頭番地のアドレス入力待ちになります。

  FROM ×××× − *

 c. アドレスをキー入力して，スペースを入力すると，記録するメモリー最終番地のアドレス※※の入力待ちになります。

  TO △△△△ − *

 d. アドレスをキー入力したところで，テープレコーダーを録音状態にして走行スタートします。テープがスタートしたら，スペースを入力します。

 注. c項，d項のとき，スペースキーの代りに [RETURN] キーを入力するとモニターのコマンド待ちに戻ります。

 e. d項のスペース入力によって記録が開始され，ファイル名とブロック番号が表示されます。記録が終了すると，モニターのコマンド待ちに戻ります。テープを停止してください。

*Pコマンドで記録したテープは，Vコマンドにより正しく記録されているかどうかを確認してください。

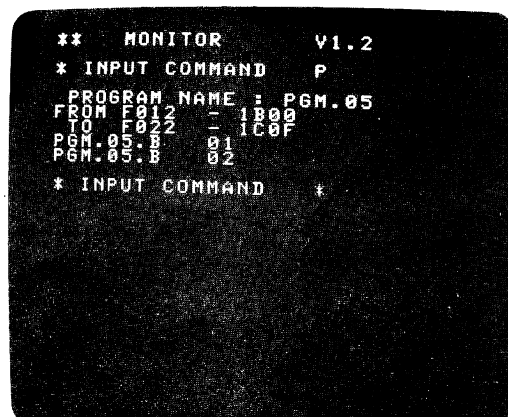
*d項により記録を開始したあと，何らかの理由により記録を中断したいときは，[BREAK RESET] キーを押してください。

※ファイル名の途中にスペースを入れることはできません。

※※ここで入力するアドレスは，記録する最終番地のアドレス＋1番地のアドレスにしてください。

使用例

1. $1B00番地から，$1C0F番地までの“PGM. 05”というファイル名のプログラムを記録しました。



2. V(VERIFY)コマンド

1. 機能

a．テープに記録されている内容と，メモリーの内容とを比較し，メモリーの内容がテープに正しく記録されているかを確認します。

2. 操作

a．モニターのコマンド待ちの状態で[V]をキー入力すると，Ｖコマンドが起動し，ファイル名の入力待ちとなります。このとき[RETURN]キーをおすと，Ｖコマンドが実行されるので注意してください。

　ＰＲＯＧＲＡＭ　ＮＡＭＥ　：　＊

b．テープを確認するプログラムが記録されている個所の前まで巻戻しておきます。

c．Ｐコマンドで記録したときのファイル名をキー入力し，スペースまたは，[RETURN]を入力すると，Ｖコマンドの実行が開始されます。

d．テープを再生走行させます。

e．プログラムが正しく記録され，かつ正しく再生されたときは，Ｐコマンドのとき同様，ファイル名とブロック番号が表示されます。

f．メモリーの内容と，テープに記録された内容とが一致しない場合は，ファイル名のあとに　＊ＥＲＲＯＲ　と表示され，警告音を発します。

　＊ＥＲＲＯＲが発生したときは，ＢＡＳＩＣのプログラムの記録・再生の項，９７ページ以降をご参照のうえ，処置してください。

g．Ｖコマンドが終了すると，モニターのコマンド待ちに戻ります。

＊ｃ項によりＶコマンドを開始したあと，何らかの理由により中断したいときは，[BREAK RESET]キーを押してください。

使用例

1. "ＰＧＭ．０５"をＶＥＲＩＦＹしているところです。

```
** MONITOR        V1.2
* INPUT COMMAND   P
 PROGRAM NAME : PGM.05
FROM F012  - 1B00
 TO  1C0F  - 1C0F
PGM.05.B    01
PGM.05.B    02
* INPUT COMMAND   V
 PROGRAM NAME : PGM.05
PGM.05.B    01
PGM.05.B    02
PGM.05.B
* INPUT COMMAND    *
```

3. L(LOAD)コマンド

1. 機能

a. テープに記録された6文字以内のファイル名で定義されたプログラムをベーシックマスタージュニア本体のメモリーに再生格納します。

2. 操作

a. モニターのコマンド待ちの状態で[L]をキー入力すると，Lコマンドが起動し，ファイル名の入力待ちとなります。このとき[RETURN]キーをおすと，Lコマンドの実行がスタートし，ファイル名をキー入力することができません。もし，ファイル名をキー入力する前に[RETURN]を誤っておしたときは BREAK RESET[ ] キーをおして実行を中断し，最初からやり直してください。

PROGRAM NAME ： *

b. テープを再生格納するプログラムが記録されている個所の前まで巻戻しておきます。

c. Pコマンドで記録したときのファイル名をキー入力し，スペースまたは，[RETURN]を入力すると，Lコマンドの実行が開始されます。

d. テープを再生走行させます。

e. プログラムがメモリーに正しく格納されたときは，ファイル名とブロック番号が表示されます。

f. 正しく格納されなかったときはファイル名のあとに ＊ERROR が表示され，警告音を発生します。

＊ERRORが発生したときは，BASICのプログラムの記録・再生の項，97ページ以降をご参照のうえ処置してください。

g. Lコマンドが終了すると，モニターのコマンド待ちに戻ります。

＊テープへの記録，再生は1ブロックを256バイトに区切って行なわれますが，エラー（＊ERROR）がプログラムの一部に発生し，正常に格納されているブロックがあるときは，エラーの発生したブロックだけを再度LOADしなおせば十分です。

使用例

1. "PGM. 05"をLOADしているところです。

```
**   MONITOR       V1.2
* INPUT COMMAND    L
 PROGRAM NAME : PGM.05
PGM.05.B   01
PGM.05.B   02
PGM.05.B
* INPUT COMMAND    *
```

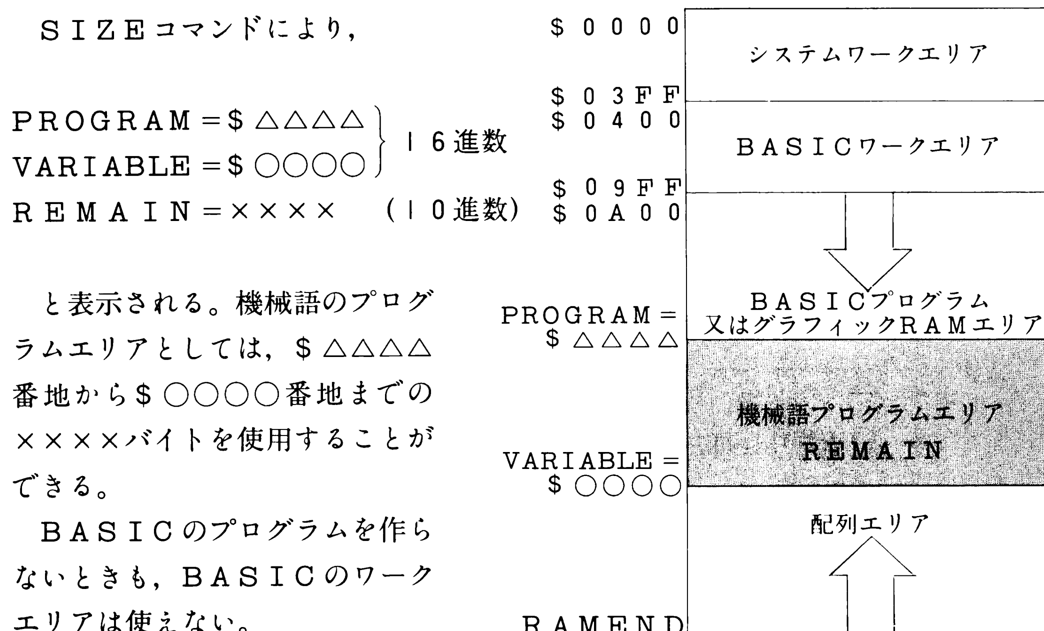
4.4.機械語について／●機械語プログラムの格納エリア

ベーシックマスタージュニアは日立マイクロコンピューターＨＭＣＳシリーズＨＤ４６８００を使用しており，モニターのコマンドを使用して６８００命令語体系の機械語プログラムを作り，実行させることができます。

ユーザーのプログラムエリアとしては＄０４００番地から＄３ＦＦＦ番地（標準実装）のＲＡＭが開放されていますが，ＢＡＳＩＣのプログラムにリンクさせるために機械語のプログラムを作るときは，ＢＡＳＩＣのプログラムが格納されているエリアをＳＩＺＥコマンドによって確認してから，プログラムを格納するアドレスを決定してください。

また，ＢＡＳＩＣのプログラムは作らず，機械語だけのプログラムを作る場合にも，＄４００番地から＄９ＦＦ番地まではＢＡＳＩＣのダイレクト実行，コマンドなどのワークエリアとして使っているため，使用することは出来ません。もし，このエリアにプログラムを格納した場合，ＲＥＳＥＴなどでやむを得ずＢＡＳＩＣに制御が移ったとき，モニターにＭＯＮＩＴＯＲコマンドで戻るときにプログラムが破壊されることがあります。

また，グラフィック１を指定した場合は＄Ａ００番地〜＄２１ＦＦ番地，グラフィック２を指定した場合は＄２１００番地〜＄３９ＦＦ番地を使用することが出来ません。グラフィックが使えないばかりでなく，グラフィック指定の際プログラムが破壊されます。



（ＲＡＭＥＮＤはＲＡＭ16Ｋバイト標準実装で，＄３ＦＦＦ番地。）

実際にプログラムのエリアを決める場合は，ＳＩＺＥコマンドにより残りのＲＡＭエリアを調べたあと，ＢＡＳＩＣのプログラムの変更，追加などの余裕をとり，端数のない覚え易い番地からスタートするようにすればよいでしょう。また，よく使うようなサブルーチンを機械語で作る場合，リロケータブルなプログラムにしておけば，ベーシックマスタージュニアのＴコマンドや，モニターＲＯＭの転送（ＭＯＶＢＬＫ）サブルーチンを使って必要に応じて転送し，メモリーを効率よく使用することができます。

ＢＡＳＩＣのＮＥＷコマンドや，ＲＥＳＥＴによって機械語のプログラムが消えることはありません。

●機械語のプログラムについて

機械語でプログラムを作る場合，全ての作業を機械語で作るのは必ずしも容易ではありません。また，ＢＡＳＩＣという高級言語がすでに備っているのに，全てのプログラムを機械語で作るのは無駄でもあり，機械語のプログラムはＢＡＳＩＣのプログラムを補間する目的で，特に機械語でプログラムを作った方が容易な場合や，処理時間が著るしく早くなるようなルーチンだけを機械語のプログラムで作り，ＢＡＳＩＣとリンクして目的の処理を行うのが，実際的です。

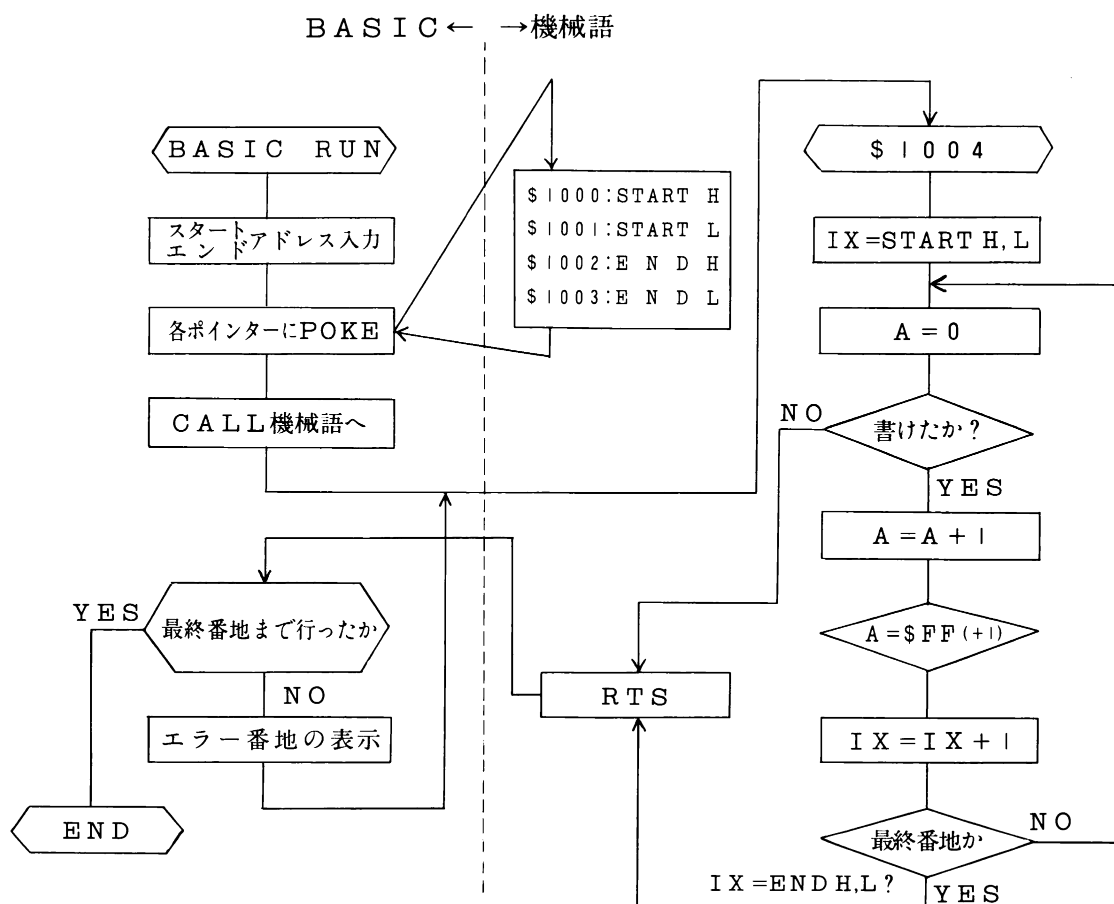
ベーシックマスタージュニアは，ＰＥＥＫ，ＰＯＫＥ，ＣＡＬＬなどのメモリーエリアを直接アクセスするステートメントをもち，また，１６進数をそのまま数値として取り扱えるため，機械語とのリンケージは容易に行なえます。

つぎに，ＢＡＳＩＣと機械語を結びつけたプログラム例を示します。このプログラムは，データ入力，ディスプレイなどの部分をＢＡＳＩＣが受けもち，直接的な作業は機械語で行なっています。機械語に取り組まれる方の参考になれば幸いです。

１．メモリーチェックプログラム。

●**機能**．ＢＡＳＩＣで入力したメモリーエリアの１つ１つのバイトに０から順次＄ＦＦまでのデータを書き込み，正しく書き込まれたかをチェックし，エラーがあれば，そのアドレスを表示する。ＢＡＳＩＣのソースプログラムの入っているエリア＄Ａ００～＄Ｃ２９の部分もチェックできる。但し，ゼロページ（＄０～＄ＦＦ），ベーシックのワークエリア（＄４００～＄９ＦＦ）及び機械語プログラムの部分（＄１０００～＄１０２８）はチェックできない。

●**概略フローチャート**



例117

機械語プログラムの部分をMコマンドで入力する。

アドレス	機械語	ラベル	オペレーター	オペランド	コメント
$1000	×× ××	START	RMB	2	スタート（IXセイブエリア）
$1002	×× ××	END	RMB	2	エンド
$1004	FE 10 00	LOOP1	LDX	START	IX←$1000
$1007	86 00		LDAA	#00	0⟶ACCA
$1009	E6 00		LDAB	0, X	メモリ⟶ACCBに退避
$100B	A7 00	LOOP2	STAA	0, X	ACCA→メモリーに書き込め
$100D	A1 00		CMPA	0, X	ACCA⟷メモリー比較せよ。
$100F	26 11		BNE	RETN2	等しくなかったらRETN2へ。
$1011	4C		INCA		A=A+1
$1012	26 F7		BNE	LOOP2	0～$FF～0までくり返し。
$1014	E7 00		STAB	0, X	ACCB メモリーに戻す。
$1016	BC 10 02		CPX	END	終了番地までチェックしたか。
$1019	27 06		BEQ	RETN1	終ったらRTSへ。
$101B	08		INX		終ってなければIX=IX+1
$101C	FF 10 00		STX	START	IXを$1000番地に書き込め。
$101F	20 E3		BRA	LOOP1	次のバイトチェック。
$1021	39	RETN1	RTS		BASICに戻れ。
$1022	E7 00	RETN2	STAB	0, X	ACCBをメモリーに戻せ。
$1024	08		INX		IX=IX+1
$1025	FF 10 00		STX	START	IXを$1000番地に格納。
$1028	39		RTS		BASICに戻れ。

BASICのプログラム（リストは全て省略形）

```
 10 E=0:CLR:!=4,4:?"*** MEMORY CHECK ***"
 20 ?:IN "␣␣␣␣START ADDRESS",F$        $100は0100で入力。
 30 IF LEN(F$)<>4 GO 10
 40 ?:IN "␣␣␣␣END ADDRESS  ",T$:?     $3FFは03FFで入力。
 50 IF LEN(T$)<>4 GO 40
 60 H1$="$"+LEFT$(F$,2)            ↑
 70 L1$="$"+RIGHT$(F$,2)           | ここまでのプログラムは
 80 H2$="$"+LEFT$(T$,2)            | 4桁の文字列(:16進表現を)
 90 L2$="$"+RIGHT$(T$,2)           | 4桁の16進数に変換して
100 P=$1000                        | POKE。
110 POKE P,VAL(H1$)                |  $1000:START
120 POKE P+1,VAL(L1$)              |  $1002:END
130 POKE P+2,VAL(H2$)              | を書き込む。
140 POKE P+3,VAL(L2$)              ↓
150 CALL P+4                         $1004:機械語スタートアドレス
160 F=PEEK(P)*256+PEEK(P+1)        } $1000,1002番地のそれぞれの
170 T=PEEK(P+2)*256+PEEK(P+3)      } 2バイトを読みとる。
180 IF (E=0)*(F=T) GO 220            エラーがない。
190 ?"____ADDRESS";HEX(F-1);"IS ERROR."
200 IF F-1<>T E=E+1:GO 150           エラーのときはINXしてRTSしている
210 ?:?"____TOTAL ERROR =";E;"BYTES":END
220 CLR:!=7,5:?"*** NO ERROR ***":END
```

● アドレス方式について

アドレス形式には，以下の7種類があります。

1. アキュムレーターアドレッシング
2. インプライドアドレッシング
3. イミディエイトアドレッシング
4. ダイレクトアドレッシング
5. エクステンディットアドレッシング
6. インデックスアドレッシング
7. リラティブアドレッシング

1. アキュムレーターアドレッシング

13種類(ASL, ASR, CLR, COM, DEC, INC, LSR, NEG, PSH, PUL, ROL, ROR, TST)の実行命令は，オペランドとしてアキュムレーターのAまたはBを指定するだけの1バイト命令です。

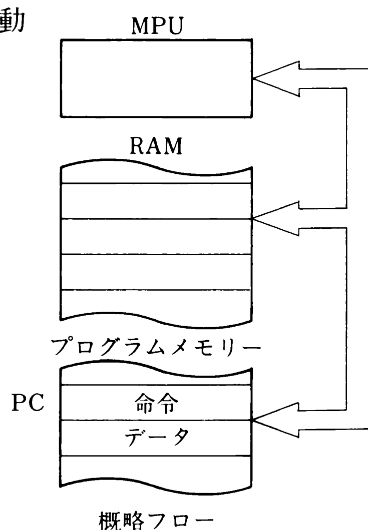
2. インプライドアドレッシング(IMPLIED)

24種類(ABA, CBA, CLC, CLI, CLV, DAA, DES, DEX, INS, INX, NOP, RTI, RTS, SBA, SEC, SEI, SEV, SWI, TAB, TAP, TBA, TPA, TSX, WAI)の実行命令は，すでにオペレーターフィールドの命令によって操作の対象が明確であり，オペランドは不要です。これらも1バイト命令です。

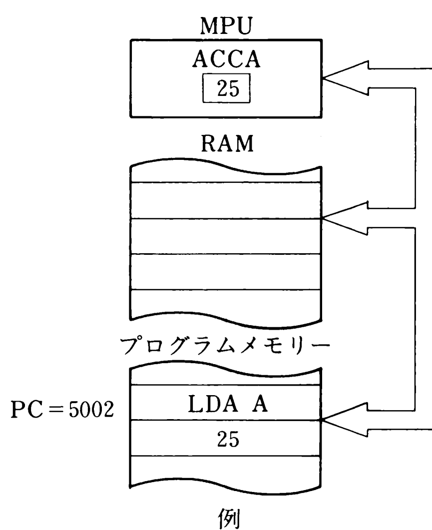
3. イミディエイトアドレッシング(IMMED)

13種類(ADC, ADD, AND, BIT, CMP, CPX, EOR, LDA, LDS, LDX, ORA, SBC, SUB)の実行命令は，オペランドの数値を直接処理することができます。CPX, LDX, LDSはオペランドが2バイトの3バイト命令ですが，その他はオペランドの数値が1バイト（0～$FFまで）の，2バイト命令となります。イミディエイトアドレッシングは，LDA　A　#$25のように，数値の前に#記号をつけて表しています。

データ移動



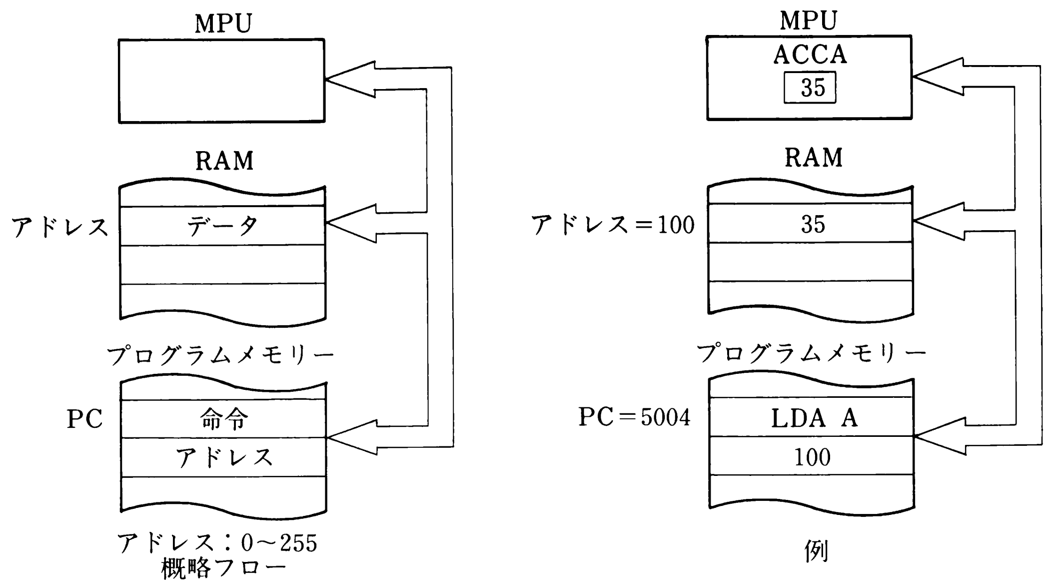
概略フロー



例

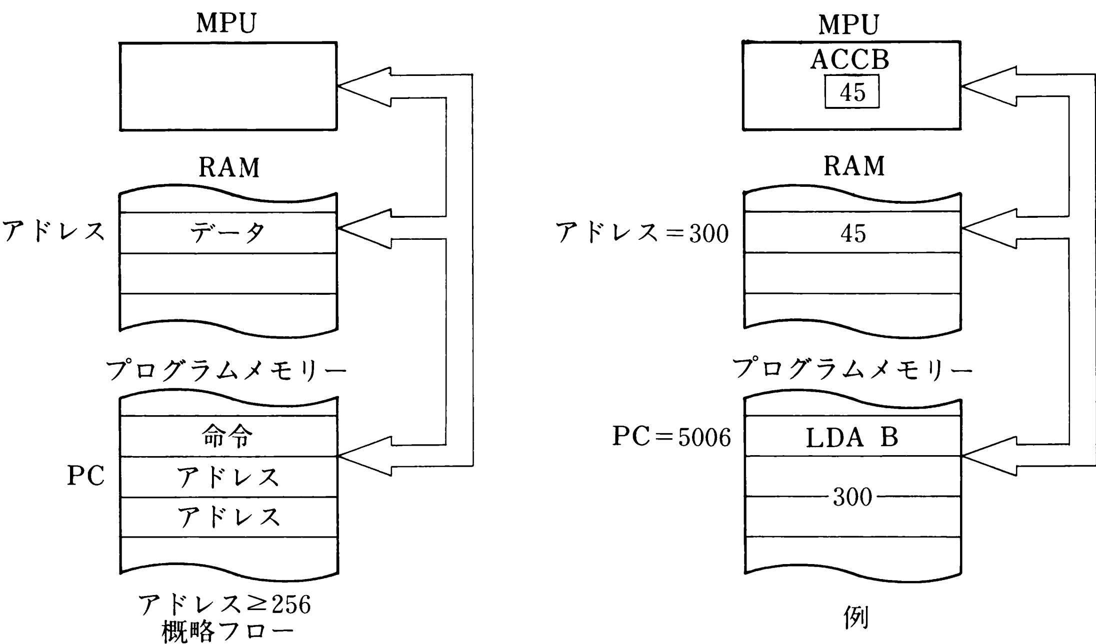
4．ダイレクトアドレッシング(DIRECT)

ダイレクトアドレッシングは，基本的にはエクステンディットアドレッシングと同じです。オペランドの示すアドレスが＄０００0～＄００ＦＦのときは，上位2バイトの００を省略することができるため，2バイト命令となり，メモリーを節約することができます。なお，ベーシックマスタージュニアは，ＲＡＭの＄００～＄ＦＦ番地をシステムのワークエリアとして使っているため，ユーザーがダイレクトアドレッシングを使用する場合は，159ページに示した内容の操作しか出来ませんので，ご注意ください。



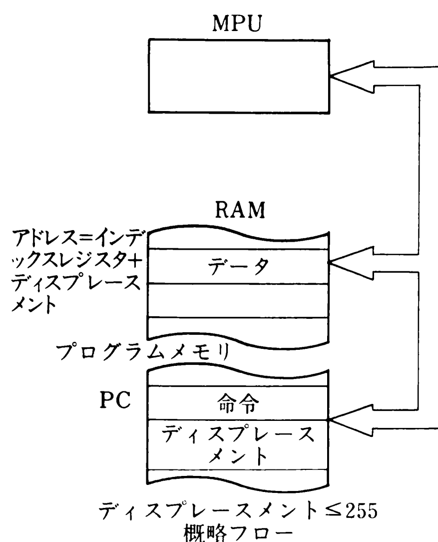
5．エクステンディッドアドレッシング(EXTND)

メモリーの絶対番地がオペランドになり，メモリーの内容により操作されます。

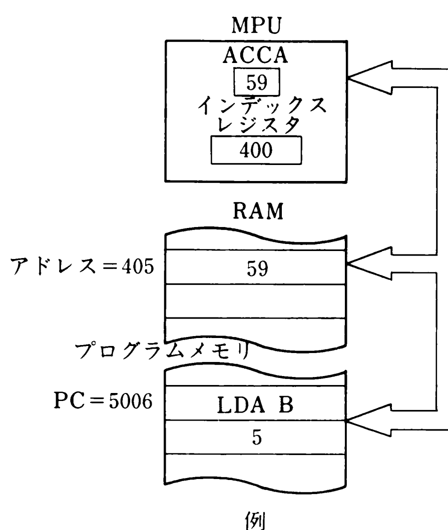


6. インデックスアドレッシング(INDEX)

インデックスレジスターの内容（2バイト）により参照するメモリーのアドレスを操作の対象とします。オペランドにはインデックスレジスターに加算する1バイトのオフセット値が入り，実際の対象となるアドレスはインデックスレジスターにオフセット値を加算したものになります。2バイト命令です。



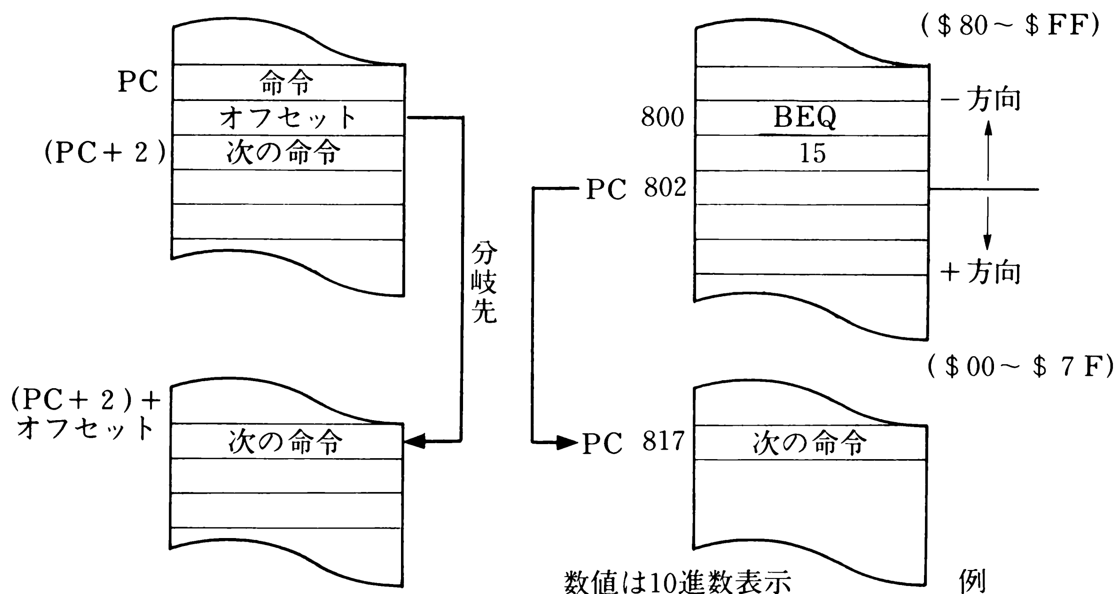
ディスプレースメント≦255
概略フロー



例

7. リラティブアドレッシング(RELATIVE)

リラティブアドレッシングは分岐命令（BCC，BCS，BEQ，BGE，BGT，BHI，BLE，BLS，BMI，BNE，BPL，BRA，BSR，BVC，BVS BLT）のときに適用され，分岐先はブランチ命令を行なうアドレスからの相対的な距離で表わされます。オペランドには1バイトの数値が入り，その数値は2の補数として取り扱われます。すなわち，＄００～＄ＦＦは＋１２７～－１２８までの数値となり，ＰＣとの距離を示します。このときＰＣの値とは，ブランチ命令のあるアドレスに２を加えたものです。



数値は10進数表示　　例

●命令一覧表

命令一覧表に用いられる記号について説明します。

記　号	説　　　　　明	記　号	説　　　　　明
(　)	(　)の内の，内容を表わす。	R	常にリセットされる。
←	矢印の方向に転送。	S	常にセットされる。
↑	スタックからもってくる。		
↓	スタックへ入れる。	①	もし，0から減算した結果に桁借りがあれば，ＣＢｉｔは，ＡＣＣＸあるいはＭの内容が$(00)_{16}$の時を除き，総ての場合セットされる。
・	論理積をとる。(ＡＮＤ)		
⊙	論理和をとる。(ＯＲ)		
⊕	排他的論理和をとる。(ＥＸＯＲ)		
		②	もし，0から減算した結果に2の補数のオーバフローがあれば，セットされる。これは，ＡＣＣＸ或るいはＭの内容が$(80)_{16}$の時に起る。
−	通常の算術上のマイナスを表わす。		
＋	通常の算術上のプラスを表わす。		
Msp	スタックポインタで示されるメモリの内容。	③	ＢＣＤの加算をした場合と同様な規則に相当しセットあるいはリセットされる。
ＡＣＣＸ	アキュムレーターＡあるいはＢを表わす。	④	もし，操作の結果，2の補数のオーバフローがあればセット，さもなくばリセット。操作前にＡＣＣＸとＭの内容が，$(80)_{16}$の時だけ，2の補数のオーバフローが起る。
ＡＣＣＡ	アキュムレーターＡを表わす。		
ＡＣＣＢ	アキュムレーターＢを表わす。		
ＣＣ	コンディションコードレジスタを表わす。		
ＩＸ	インデックスレジスタの16Ｂｉｔを表わす。		
ＩＸＬ	インデックスレジスタの下位8Ｂｉｔを表わす。	⑤	もし，操作の結果，2の補数のオーバフローがあればセット，さもなくばリセット。操作前にＡＣＣＸ或るいはＭの内容が，$(7F)_{16}$の時だけ，2の補数のオーバフローが起る。
ＩＸＨ	インデックスレジスタの上位8Ｂｉｔを表わす。		
ＰＣ	プログラムカウンタの16Ｂｉｔを表わす。		
ＰＣＬ	プログラムカウンタの下位8Ｂｉｔを表わす。		
ＰＣＨ	プログラムカウンタの上位8Ｂｉｔを表わす。		
ＳＰ	スタックポインタの16Ｂｉｔを表わす。	⑥	もし，操作の結果，〝ＮがセットでＣがクリア″かあるいは〝ＮがクリアでＣがセット″ならばセット，さもなくばクリア。
ＳＰＬ	スタックポインタの下位8Ｂｉｔを表わす。		
ＳＰＨ	スタックポインタの上位8Ｂｉｔを表わす。		
	コンディションコード		
ＣＢｉｔ	ＣＣの桁上がり，借り表示のＢｉｔ。	⑦	ＩＸの上位8Ｂｉｔからの減算の結果，最上位Ｂｉｔがセットならばセット，さもなくばクリア。
ＶＢｉｔ	ＣＣの2の補数のオーバフロー表示のＢｉｔ。		
ＺＢｉｔ	ＣＣのゼロ表示のＢｉｔ。		
ＮＢｉｔ	ＣＣのマイナス表示のＢｉｔ。	⑧	ＩＸの上位8Ｂｉｔからの減算の結果，上位8Ｂｉｔから2の補数のオーバフローが起きたならばセット，さもなくばクリア。
ＩＢｉｔ	ＣＣの割込みマスク表示のＢｉｔ。		
ＨＢｉｔ	ＣＣのハーフキャリ表示のＢｉｔ。		
Ｍ	記憶装置(メモリー)の番地。(1バイト)	⑨	操作の結果，ＩＸの最上位Ｂｉｔがセットされればセット，さもなくばクリア。
Ｍ＋1	Ｍで表示された記憶装置の番地に0001だけ加算された番地の1バイト，すなわちＭの次の番地。	⑩	スタックから引き出された状態で再格納される。
		⑪	割込要求信号が割込要求制御線上に検出されるまで影響なし，それ以上の実行は，ＩＢｉｔが始めリセットされていれば，割込要求を受けた時，ＩＢｉｔをセットした上で行う。
ＯＰ	16進表示の機械語命令。		
＃	機械語のバイト数。		
●	影響なし。		
○	真ならばセットされ，さもなくばクリアされる。	⑫	ＡＣＣＸのＢｉｔ0〜5の状態により決まる。

●命令と働き

ADC, ADD, AND, BIT, CMP, EOR

操作記号	操作および操作の説明	アドレス方式								CCレジスター
		IMMED		DIRECT		INDEX		EXTEND		
		OP	#	OP	#	OP	#	OP	#	HINZVC
	ADd with Carry									
ADC	C Bit の内容をＡＣＣＸとＭの内容の合計に加え，結果をＡＣＣＸに置く。									
ADCA	A←(A)+(M)+(C)	8 9	2	9 9	2	A 9	2	B 9	3	○●○○○○
ADCB	B←(B)+(M)+(C)	C 9	2	D 9	2	E 9	2	F 9	3	○●○○○○
	ADD without carry									
ADD	ＡＣＣＸの内容にＭの内容を加えて，結果をＡＣＣＸに置く。									
ADDA	A←(A)+(M)	8B	2	9 B	2	AB	2	BB	3	○●○○○○
ADDB	B←(B)+(M)	CB	2	DB	2	EB	2	FB	3	○●○○○○
	Logical AND									
AND	ＡＣＣＸの内容とＭの内容の各Ｂｉｔ間で論理積をとり，結果をＡＣＣＸに置く。									
ANDA	A←(A)・(M)	8 4	2	9 4	2	A 4	2	B 4	3	●●○○R●
ANDB	B←(B)・(M)	C 4	2	D 4	2	E 4	2	F 4	3	●●○○R●
	BIt Test									
BIT	ＡＣＣＸの内容とＭの内容の各Ｂｉｔ間で論理積をとり，結果により状態コードを修正する。									
BITA	(A)・(M)	8 5	2	9 5	2	A 5	2	B 5	3	●●○○R●
BITB	(B)・(M)	C 5	2	D 5	2	E 5	2	F 5	3	●●○○R●
	CoMPare									
CMP	ＡＣＣＸの内容とＭの内容を比較しＣＣレジスターを決定する。									
CMPA	(ACCA)−(M)	8 1	2	9 1	2	A 1	2	B 1	3	●●○○○○
CMPB	(ACCB)−(M)	C 1	2	D 1	2	E 1	2	F 1	3	●●○○○○
	Exclusive OR									
EOR	ＡＣＣＸの内容とＭの内容の各Ｂｉｔ間で排他的論理和をとり，結果をＡＣＣＸに置く。									
EORA	A←(A)⊕(M)	8 8	2	9 8	2	A 8	2	B 8	3	●●○○R●
EORB	B←(B)⊕(M)	C 8	2	D 8	2	E 8	2	F 8	3	●●○○R●

命令の実行に必要なマシンサイクルについては省略しています。

LDA, ORA, STA, SBC, SUB, ABA, CLR

操作記号	操作及び操作の説明	アドレス方式 IMMED OP	#	DIRECT OP	#	INDEX OP	#	EXTND OP	#	CCレジスター HINZVC
	LoaD Accumulator									
LDA	メモリーの内容をACCXにロードする。									
LDAA	A←(M)	86	2	96	2	A6	2	B6	3	●●○○R●
LDAB	B←(M)	C6	2	D6	2	E6	2	F6	3	●●○○R●
	Inclusive OR									
ORA	ACCXの内容とMの内容の各Bit間で論理和をとり，結果をACCXに置く。									
ORAA	A←(A)⊙(M)	8A	2	9A	2	AA	2	BA	3	●●○○R●
ORAB	B←(B)⊙(M)	CA	2	DA	2	EA	2	FA	3	●●○○R●
	STore Accumulator									
STA	ACCXの内容をメモリーに格納する。ACCXの内容は変化せずに残る。									
STAA	M←(ACCA)			97	2	A7	2	B7	3	●●○○R●
STAB	M←(ACCB)			D7	2	E7	2	F7	3	●●○○R●
	SuBtract with Carry									
SBC	ACCXの内容からMとCの内容を減算して，結果をACCXに置く。									
SBCA	ACCA←(ACCA)−(M)−(C)	82	2	92	2	A2	2	B2	3	●●○○○○
SBCB	ACCB←(ACCB)−(M)−(C)	C2	2	D2	2	E2	2	F2	3	●●○○○○
	SUBtract									
SUB	ACCXの内容からMの内容を減算して，結果をACCXに置く。									
SUBA	ACCA←(ACCA)−(M)	80	2	90	2	A0	2	B0	3	●●○○○○
SUBB	ACCB←(ACCB)−(M)	C0	2	D0	2	E0	2	F0	3	●●○○○○

		INDEX OP	#	EXTND OP	#	IMPLIED OP	#	CCレジスター
	Add accumulator B to accumulator A							
ABA	ACCBの内容をACCAの内容に加えて，結果をACCAに置く。							
	ACCA←(ACCA)+(ACCB)					1B	1	○●○○○○
	CLeaR							
	ACCXあるいはMの内容が0に置き換られる。							
CLR	M←00	6F	2	7F	3			●●RSRR
CLRA	ACCA←00					4F	1	●●RSRR
CLRB	ACCB←00					5F	1	●●RSRR

操作記号	操作及び操作の説明	アドレス方式						CCレジスター
		INDEX		EXTND		IMPLIED		HINZVC
		OP	#	OP	#	OP	#	
	Compare accumulators							
CBA	ACCAの内容とACCBの内容を比較し，結果に従ってCCレジスタを設定する。							
	(ACCA)−(ACCB)					11	1	●●○○○○
	COMplement							
	ACCXの内容あるいはMの内容が，1の補数に置き換えられる。							
COM	M←FF₁₆−(M)	63	2	73	3			●●○○RS
COMA	ACCA←FF₁₆−(ACCA)					43	1	●●○○RS
COMB	ACCB←FF₁₆−(ACCB)					53	1	●●○○RS
	NEGate							
	ACCXの内容あるいはMの内容が，2の補数に置き換えられる。							
NEG	M←00−(M)	60	2	70	3			●●○○①②
NEGA	ACCA←00−(ACCA)					40	1	●●○○①②
NEGB	ACCB←00−(ACCB)					50	1	●●○○①②
	(注)：ただし80₁₆は置き換えられず残る。							
	DECrement							
	ACCXの内容あるいはMの内容から1を減算する。							
DEC	M←(M)−01	6A	2	7A	3			●●○○④●
DECA	ACCA←(ACCA)−01					4A	1	●●○○④●
DECB	ACCB←(ACCB)−01					5A	1	●●○○④●
	INCrement							
	ACCXの内容あるいはMの内容に1を加算する。							
INC	M←(M)+01	6C	2	7C	3			●●○○⑤●
INCA	ACCA←(ACCA)+01					4C	1	●●○○⑤●
INCB	ACCB←(ACCB)+01					5C	1	●●○○⑤●
	PuSH data onto stack							
PSH	ACCXの内容は，SPの示すアドレスのメモリーに格納される。その時SPは1減算される。							
PSHA	Msp←(ACCA), SP←SP−0001					36	1	●●●●●●
PSHB	Msp←(ACCB), SP←SP−0001					37	1	●●●●●●
	PULℓ data from stack							
PUL	SPは1加算される。その後SPにあるアドレスのメモリーからACCXにロードされる。							
PULA	SP←SP−0001, ACCA←Msp					32	1	●●●●●●
PULB	SP←SP−0001, ACCB←Msp					33	1	●●●●●●

ROL, ROR, ASL, ASR, LSR

操作記号	操作及び操作の説明	アドレス方式 INDEX OP	INDEX #	EXTND OP	EXTND #	IMPLIED OP	IMPLIED #	CCレジスター HINZVC
	ROtate Left ACCXの内容あるいはMの内容の全Bitを左に1桁だけ桁送りする。Bit 0はC Bitよりロードされ C BitはACCXあるいはMの最上位Bitよりロードされる。							
ROL	M	69	2	79	3			●●○○⑥○
ROLA	ACCA　C ← Bit 7 … Bit 0					49	1	●●○○⑥○
ROLB	ACCB　C Bit　Bit 7　Bit 0					59	1	●●○○⑥○
	ROtate Right ACCXの内容あるいはMの内容の全Bitを右に1桁だけ桁送りする。Bit 7はC Bitよりロードされ C BitはACCXあるいはMの最下位Bitよりロードされる。							
ROR	M	66	2	76	3			●●○○⑥○
RORA	ACCA　C → Bit 7 … Bit 0					46	1	●●○○⑥○
RORB	ACCB　Bit 7　Bit 0　0					56	1	●●○○⑥○
	Arithmetic Shift Left ACCXの内容あるいはMの内容の全Bitを左に1桁だけ桁送りする。Bit 0は0がロードされ，C BitはACCX又はMの最上位Bitからロードされる。							
ASL	M	68	2	78	3			●●○○⑥○
ASLA	ACCA　C ← Bit 7 … Bit 0 ← 0					48	1	●●○○⑥○
ASLB	ACCB　C Bit　Bit 7　Bit 0					58	1	●●○○⑥○
	Arithmetic Shift Right ACCXの内容あるいはMの内容の全Bitを右に1桁だけ桁送りする。ただしBit 7はそのまま保持され，Bit 0はC Bitへロードされる。							
ASR	M	67	2	77	3			●●○○⑥○
ASRA	ACCA　Bit 7 … Bit 0 → C					47	1	●●○○⑥○
ASRB	ACCB　Bit 7　Bit 0　C Bit					57	1	●●○○⑥○
	Logical Shift Right ACCXの内容あるいはMの内容の全Bitを右に1桁だけ桁送りする。Bit 7は0がロードされ C BitはACCXあるいはMの最下位Bitからロードされる。							
LSR	M	64	2	74	3			●●R○⑥○
LSRA	ACCA　0 → Bit 7 … Bit 0 → C					44	1	●●R○⑥○
LSRB	ACCB　Bit 7　Bit 0　C Bit					54	1	●●R○⑥○

操作記号	操作及び操作の説明	アドレス方式 INDEX OP	#	EXTND OP	#	IMPLIED OP	#	CCレジスター HINZVC
SBA	SuBtract Accumurator ACCAの内容からACCBの内容を減算し結果をACCAに置く。 ACCA←(ACCA)−(ACCB)					10	1	●●○○○
TAB	Transfer from accumurator A to accumurator B ACCAの内容をACCBに移す，ACCBの前の内容は失われる。 ACCB←(ACCA)					16	1	●●○○R●
TBA	Transfer from accumurator B to accumurator A ACCBの内容をACCAに移す，ACCAの前の内容は失われる。 ACCA←(ACCB)					17	1	●●○○R●
TST	TeST ACCXの内容あるいはMの内容に相当するコンディションコードのNとZを設定する。 M←(M)−00	6D	2	7D	3			●●○○RR
TSTA	ACCA←(ACCA)−00					4D	1	●●○○RR
TSTB	ACCB←(ACCB)−00					5D	1	●●○○RR
DAA	Decimal Adjust ACCA もし，ACCXの内容と，C Bit，H Bitの状態が，BCD数値に加算操作(ABA，ADD，ADC命令)を行った結果であるならば，DAA操作は，次の様になる。前記の条件を仮定して，DAA操作は，ACCAの内容とC Bitを，正しいBCDの和と正しいC Bitの表示に合せる。 ┌→ACCA └(ACCA)+(00，06，60or66) 上記の操作を行い，C Bitをセットする。					19	1	●●○○○③

操作記号	操作及び操作の説明	IMMED OP	#	DIRECT OP	#	INDEX OP	#	EXTND OP	#	CCレジスター HINZVC
CPX	ComPare indeX register IXレジスタの上位バイトの内容は，プログラムで示されるアドレスのメモリー内容と比較される。IXレジスターの下位バイトの内容は，プログラムで示されるアドレスの次のアドレスのメモリーと比較される。 (IXH)−(M) (IXL)−(M+1)	8C	3	9C	2	AC	2	BC	3	●●⑦○⑧●

LDX, DEX, DES, INX, INS, TXS, TSX

操作記号	操作及び操作の説明	アドレス方式								CCレジスター
		IMMED		DIRECT		INDEX		EXTND		HINZVC
		OP	#	OP	#	OP	#	OP	#	
LDX	LoaD indeX register プログラムで示されるアドレスのメモリー内容(1バイト)を，IXレジスターの上位バイトにロードする。プログラムで示されるアドレスの次のアドレスのメモリー内容(1バイト)をIXレジスターの下位バイトにロードする。 IXH←(M) IXL←(M+1)	CE	3	DE	2	EE	2	FE	3	●●⑨○R●

操作記号	操作及び操作の説明	IMPLIED		CCレジスター
		OP	#	HINZVC
DEX	DEcrement indeX registor IXレジスターから1を減算する IX←(IX)−0001	09	1	●●●○●●
DES	DEcrement Stack pointer SPから1を減算する。 SP←(SP)−0001	34	1	●●●●●●
INX	INcremnt IndeX register IXレジスターに1を加算する。 IX←(IX)+0001	08	1	●●●○●●
INS	INcrement Stack pointer SPに1を加算する。 SP←(SP)+0001	31	1	●●●●●●
TXS	Transfer from indeX registor to Stack pointer IXレジスターの内容から1を減算しSPにロードする。IXレジスターの内容は変らない。 SP←(IX)−0001	35	1	●●●●●●
TSX	Transfer from Stack pointer to indeX registor SPの内容に1を加えIXレジスターにロードする。SPの内容は変らない。 IX←(SP)+0001	30	1	●●●●●●

操作記号	操作及び操作の説明	アドレス方式 IMMED OP	#	DIRECT OP	#	INDEX OP	#	EXTND OP	#	CCレジスター HINZVC
LDS	LoaD Stack pointer プログラムで示されるアドレスのメモリー内容(1バイト)をSPの上位バイトにロードする。プログラムで示されるアドレスの次のアドレスのメモリー内容(1バイト)を，SPの下位桁にロードする。 SPH←(M) SPL←(M+1)	8E	3	9E	2	AE	2	BE	3	●●⑨○R●
STS	STore Stack pointer SPの上位バイトの内容をプログラムで示されるアドレスのメモリーに格納し，SPの下位バイトの内容をプログラムで示されるアドレスの次のアドレスのメモリーに格納する。 M←(SPH)，M+1←(SPL)			9F	2	AF	2	BF	3	●●⑨○R●
STX	STore indeX register IXレジスターの上位バイトの内容を，プログラムで示されるアドレスのメモリーに格納し，IXレジスターの下位バイト内容をプログラムで示されるアドレスの次のアドレスのメモリーに格納する。 M←(IXH)，M+1←(IXL)			DF	2	EF	2	FF	3	●●⑨○R●

操作記号	操作及び操作の説明	RELATIVE OP	#	CCレジスタ HINZVC
BRA	BRanch Always 次の公式で，与えられるアドレスに無条件分岐する，Relは相対アドレスで，分岐命令に相当する機械語の2番目のバイトに2の補数の形で記憶されている。 PC←(PC)+0002+Rel	20	2	●●●●●●
BCC	Branch if Carry Clear CCレジスタのC Bitの状態をテストし，もしC Bitがクリアであれば分岐を起す。分岐の行き先はBRAの説明に同じ。 PC←(PC)+0002+Rel, if (C)=0	24	2	●●●●●●
BCS	Branch if Carry Set CCレジスターのC Bitの状態をテストし，もしC Bitがセットであれば分岐を起す。分岐の行き先はBRAの説明に同じ。 PC←(PC)+0002+Rel, if (C)=1	25	2	●●●●●●

BEQ, BGE, BGT, BHI

操作記号	操作及び操作の説明	アドレス方式 RELATIVE OP	#	CCレジスター HINZVC
BEQ	Branchi if EQual CCレジスターのZ Bitの状態をテストし，もしZ Bitがセットであれば分岐を起す。分岐の行き先はBRAの説明に同じ。 PC←(PC)+0002+Rel, if (Z)=1	27	2	●●●●●●
BGE	Branch if Greater than or Equal zero もし，CCレジスターの"N BitがセットでV Bitがセット"かあるいは"N BitがクリアでV Bitがクリア"のどちらか一方であるならば，分岐を起す。分岐の行き先は，BRAの説明に同じ。 2の補数で表わされる被減数(即ちACCXの内容が)，2の補数で表わされる減数(即ちMの内容)より大きいかあるいは等しいという条件時において，もし，CBA，CMP，SBA，SUB等の命令直後にBGE命令が実行されたならば，分岐が起る。 PC←(PC)+0002+Rel, if (N)⊕(V)=0 即ち，もし (ACCX)≧(M) (2の補数)	2C	2	●●●●●●
BGT	Branch if Greater Than zero もし，CCレジスターの Z Bit がクリアで，しかも"N BitがセットでV Bitがセット" かあるいは"N BitがクリアでV Bitがクリア"のどちらか一方であるならば，分岐を起す。分岐の行き先は，BRAの説明に同じ。 2の補数で表わされる被減数(即ちACCXの内容)が，2の補数で表わされる減数(即ちMの内容)より大きいという条件時において，もし，CBA，CMP，SBA，SUB等の命令直後にBGE命令が実行されたならば，分岐が起る。 PC←(PC)+002+Rel, if (Z)⊙[(N)⊕(V)]=0 即ち，もし (ACCX)>(M) (2の補数)	2E	2	●●●●●●
BHI	Branch if HIgher もし，CCレジスターのC Bitがクリアで，しかもZ Bitがクリアならば，分岐を起す。 2進数の絶対値で示される被減数(即ちACCXの内容)が2進数の絶対値で示される減数(即ちMの内容)より大きいという条件時において，もしCBA，CMP，SBA，SUB等の命令直後にBHI命令が実行されたならば分岐が起る。分岐の行く先は，BRAの説明に同じ。 PC←(PC)+0002+Rel, if (C)⊙(Z)=0 即ち，もし (ACCX)>(M) (2進数の絶対値)	22	2	●●●●●●

操作記号	操作及び操作の説明	アドレス方式		CCレジスター
		RELATIVE		
		OP	#	HINZVC
BLE	Branch if Less than or Equal to zero もし，CCレジスターのZ Bitがセットで，しかも〝N BitがセットでV Bitがクリア〞かあるいは〝N BitがクリアでV Bitがセット〞のどちらか一方であるならば，分岐を起す。分岐の行く先は，BRAの説明に同じ。 2の補数で示され，被減数(即ちACCXの内容)が2の補数で示される減数(即ちMの内容)より小さいか，あるいは等しいという条件時において，もし，CBA，CMP，SBA，SUB等の命令直後にBLE命令が実行されたならば，分岐が起る。 PC←(PC)+0002+Rel, if (Z)⊙〔(N)⊕(V)〕=1 即ち，もし (ACCX)≦(M) (2の補数)	2F	2	●●●●●●
BLS	Branch if Lower or Same もし，CCレジスターのC BitがセットかあるいはZ Bitがクリアならば，分岐が起る。 2進数の絶対値で示される被減数(即ちACCXの内容)が2進数の絶対値で示される減数(即ちMの内容)より小さいかあるいは等しいという条件時において，もし，CBA，CMP，SBA，SUB等の命令直後にBLS命令が実行されたならば，分岐が起る。分岐の行く先は，BRAの説明に同じ。 PC←(PC)+0002+Rel, if (C)⊙(Z)=1 即ち，もし (ACCX)≦(M) (2進数の絶対値)	23	2	●●●●●●
BLT	Branch if Less Than zero もし，CCレジスターの〝N BitがセットでV Bitがクリア〞かあるいは，〝N BitがクリアでV Bitがセット〞のどちらか一方であるならば，分岐を起す。分岐の行く先は，BRAの説明に同じ。 2の補数で示される被減数(即ちACCXの内容)が2の補数で示される減数(即ちMの内容)より小さいと云う条件時において，もし，CBA，CMP，SBA，SUB等の命令直後にBLT命令が実行されたならば，分岐が起る。 PC←(PC)+0002+Rel, if (N)⊕(V)=1 即ち，もし (ACCX)<(M) (2の補数)	2D	2	●●●●●●
BMI	Branch if MInus CCレジスターのN Bitの状態をテストし，もし，N Bitがセットならば，分岐が起る。分岐の行く先は，BRAの説明に同じ。 PC←(PC)+0002+Rel, if (N)=1	2B	2	●●●●●●
BNE	Bran ch if Not Equal CCレジスターのZ Bitの状態をテストし，もし，Z Bitがクリアならば，分岐が起る。分岐の行く先は，BRAの説明に同じ。 PC←(PC)+0002+Rel, if (Z)=0	26	2	●●●●●●

BPL, BVC, BVS, BSR

操作記号	操作及び操作の説明	アドレス方式 RELATIVE OP	#	CCレジスター HINZVC
BPL	Branch if PLus CCレジスターのN Bitの状態をテストし，もし，N Bitがクリアならば，分岐が起る。分岐の行く先は，BRAの説明に同じ。 PC←(PC)＋0002＋Rel, if (N)＝0	2A	2	●●●●●●
BVC	Branch if oVerflow Clear CCレジスターのV Bitの状態をテストし，もし，V Bitがクリアならば，分岐が起る。分岐の行く先は，BRAの説明に同じ。 PC←(PC)＋0002＋Rel, if (V)＝0	28	2	●●●●●●
BVS	Branch if oVerflow Set CCレジスターのV Bitの状態をテストし，もし，V Bitがセットならば，分岐が起る。分岐の行く先は，BRAの説明に同じ。 PC←(PC)＋0002＋Rel, if (V)＝1	29	2	●●●●●●
BSR	Branch to SubRoutine PCは，2加算される。そのPCの内容の下位8 Bitは，スタックに入れられる。SPは次に1減算される。PCの内容の上位8 Bitは，スタックに入れられる。SPは再び1減算される。しかる後にプログラムで示された場所に分岐が起る。分岐の行く先は，BRAに同じ。 PC←(PC)＋0002 ↓ (PCL) SP←(SP)－0001 ↓ (PCH) SP←(SP)－0001 PC←(PC)＋Rel	8D	2	●●●●●●

操作記号	操作及び操作の説明	アドレス方式				CCレジスター
		INDEX		EXTND		
		OP	#	OP	#	HINZVC
JMP	JuMP オペランドに示される数値アドレスへジャンプする。その数値アドレスは，INDEXあるいはEXTNDアドレス指示方式の規則に従いジャンプ先を得る。 PC←数値アドレス	6E	2	7E	3	●●●●●●
JSR	Jump to SubRroutine PCは，アドレス指示方式の違いにより3あるいは2加算され，次にスタックに8 Bitずつ格納される。SPはスタック中の次の空白場所を示し，オペランドに示される数値アドレスへジャンプする。その数値アドレスは，INDEXあるいはEXTNDアドレス指示方式の規則に従いジャンプ先を得る。 PC←(PC)+0003(EXTNDアドレス指示) PC←(PC)+0002(INDEXアドレス指示) ↓(PCL) SP←(SP)−0001 ↓(PCH) SP←(SP)−0001 PC←数値アドレス	AD	2	BD	3	●●●●●●
				IMPLIED		CCレジスター
				OP	#	HINZVC
NOP	No Operation この命令は，1語命令で，特に操作は行わない。ただPCのみが増加され，他のレジスターは影響を受けない。 PC←(PC)+0001			01	1	●●●●●●
RTI	ReTurn from Interrupt 割込みによりスタックに退避されたレジスター（CCレジスター，ACCA，ACCB，IXレジスター，PC）の内容を再び各レジスターにロードする。 (注)：CCレジスターのI Bitは，もし，スタックの中に格納されたBitが0に相当する条件時のみ，リセットとなる。 SP←(SP)+0001，↑ CC SP←(SP)+0001，↑ACCB SP←(SP)+0001，↑ACCA SP←(SP)+0001，↑ IXH SP←(SP)+0001，↑ IXL SP←(SP)+0001，↑ PCH SP←(SP)+0001，↑ PCL			3B	1	← ⑩ →

RTS, SWI

操作記号	操作及び操作の説明	アドレス方式 IMPLIED OP	 #	CCレジスター HINZVC
RTS	ReTurn from Subroutine SPは1増加され，SPで示されるアドレスのメモリー内容（1バイト）をPCの上位8 Bitにロードする。SPは再び1増加され，SPで示されるアドレスのメモリー内容（1バイト）をPCの下位8 Bitにロードする。 SP←(SP)+0001 ↑ PCH SP←(SP)+0001 ↑ PCL	39	1	●●●●●●
SWI	SoftWare Interrpt PCの内容は，1増加され，次にPC，IXレジスター，ACCA，ACCBの内容がスタックに格納される。CCレジスターのコンディションコード，H，I，N，Z，V，Cは，各々スタックの0～5のBit位置に格納され7 Bit目と6 Bit目は1の状態にセットされる。SPはデータの各バイトが，スタックに格納された後，1減算される。次に，割込みマスクBitがセットされる。PCには，その時(n－5)と(n－4)のメモリー番地（ベクターアドレス）の内容がロードされる。前記のnはアドレスの全ラインが"H"状態に相当するアドレスである。つまり，(n－5)→FFFA16(n－4)→FFFB16をいう。 PC←(PC)+0001 ↓ (PCL), SP←(SP)－0001 ↓ (PCH), SP←(SP)－0001 ↓ (IXL), SP←(SP)－0001 ↓ (IXH), SP←(SP)－0001 ↓ (ACCA), SP←(SP)－0001 ↓ (ACCB), SP←(SP)－0001 ↓ (CC), SP←(SP)－0001 I←1 PCH←(n－0005) FFFA PCL←(n－0004) FFFB	3F	1	●S●●●●

操作記号	操作及び操作の説明	アドレス方式 IMPLIED OP	#	CCレジスター HINZVC
WAI	WAit for Interrupt PCの内容は，1増加され，次にPC，IXレジスター，ACCA，ACCBの内容がスタックに格納される。CCレジスターのコンディションコードH，I，N，Z，V，Cは，各々スタックの0～5のBit位置に格納され，7 Bit目と6 Bit目は，1の状態にセットされる。SPはデータの各バイトが，スタックに格納された後，1減算される。プログラムの実行は割込みが発生するまで中断される。 I Bitがリセット状態で，割込みが発生すると，I Bit(割込みマスクBit)はセットされプログラムカウンターには，(n－7)と(n－6)のメモリー番地（ベクターアドレス）の内容がロードされる。前記のnはアドレスバスの全ラインが〝H″状態に相当するアドレスである。つまり，(n－7)→FFF8₁₆，(n－6)→FFF9₁₆をいう。 PC←(PC)+0001 ↓(PCL)　SP←(SP)－0001 ↓(PCH)　SP←(SP)－0001 ↓(IXL)　SP←(SP)－0001 ↓(IXH)　SP←(SP)－0001 ↓(ACCA)　SP←(SP)－0001 ↓(ACCB)　SP←(SP)－0001 ↓(CC)　SP←(SP)－0001	3E	1	●⑪●●●●
CLC	CLear Carry CCレジスターのC Bitをクリアする。 C Bit←0	0C	1	●●●●●R
CLI	CLear Interrupt mask CCレジスターのI Bit(割込マスク)をクリアする。これは，もし〝割込要求″制御入力が〝L″の状態で信号が送られたならば，周辺装置からマイクロプロセッサに割込サービスができる様にしている。 I Bit←0	0E	1	●R●●●●
CLV	CLear two's complemet oVerflow bit CCレジスターの2の補数のオーバフローBit(V Bit)をクリアする。 V Bit←0	0.A	1	●●●●R●
SEC	SEt Carry CCレジスターのBit Cをセットする。 C Bit←1	0D	1	●●●●●S

SEI, SEV, TAP, TPA

操作記号	操作及び操作の説明	アドレス方式 IMPLIED OP #	CCレジスター HINZVC
SEI	SEt Interrupt mask CCレジスターのI Bit(割込マスク)をセットする。プログラムの命令実行によってI Bitがクリアされるまで，マイクロプロセッサは，周辺装置からの割込サービスを禁止する。 V Bit ← 1	0F 1	●S●●●●
SEV	SEt two's complement oVerflow Bit CCレジスターの2の補数のオーバフローBit(V Bit)をセットする。 V Bit ← 1	0B 1	●●●●S●
TAP	Transfer from accumulator A to Processor condition codes registor ACCAの0〜5のBit位置の内容を，CCレジスターの対応するBitの位置へ移動を行う。ACCAの内容は変化せず，そのまま残る。 Bit 7 6 5 4 3 2 1 0 ACCA CC	06 1	← ⑫ →
TPA	Transfer from Processor condition codes registor to accumulator A CCレジスターの内容を，ACCAの0〜5 Bitの位置に相当する所へ移動を行う。ACCAのBit 7，Bit 6は1がロードされる。CCレジスターの内容は変化せず，そのまま残る。 Bit 7 6 5 4 3 2 1 0 ACCA "1" "1"	07 1	●●●●●●

機械語—ニーモニックコード早見表

下位4ビット 上位4ビット	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	•	NOP (IMP)	•	•	•	•	TAP (IMP)	TPA (IMP)	INX (IMP)	DEX (IMP)	CLV (IMP)	SEV (IMP)	CLC (IMP)	SEC (IMP)	CLI (IMP)	SEI (IMP)
1	SBA	CBA	•	•	•	•	TAB (IMP)	TBA (IMP)	•	DAA (IMP)	•	ABA (IMP)	•	•	•	•
2	BRA (REL)	•	BHI (REL)	BLS (REL)	BCC (REL)	BCS (REL)	BNE (REL)	BEQ (REL)	BVC (REL)	BVS (REL)	BPL (REL)	BMI (REL)	BGE (REL)	BLT (REL)	BGT (REL)	BLE (REL)
3	TSX (IMP)	INS (IMP)	PUL (A)	PUL (B)	DES (IMP)	TXS (IMP)	PSH (A)	PSH (B)	•	RTS (IMP)	•	RTI (IMP)	•	•	WAI (IMP)	SWI (IMP)
4	NEG (A)	•	•	COM (A)	LSR (A)	•	ROR (A)	ASR (A)	ASL (A)	ROL (A)	DEC (A)	•	INC (A)	TST (A)	•	CLR (A)
5	NEG (B)	•	•	COM (B)	LSR (B)	•	ROR (B)	ASR (B)	ASL (B)	ROL (B)	DEC (B)	•	INC (B)	TST (B)	•	CLR (B)
6	NEG (IND)	•	•	COM (IND)	LSR (IND)	•	ROR (IND)	ASR (IND)	ASL (IND)	ROL (IND)	DEC (IND)	•	INC (IND)	TST (IND)	JMP (IND)	CLR (IND)
7	NEG (EXT)	•	•	COM (EXT)	LSR (EXT)	•	ROR (EXT)	ASR (EXT)	ASL (EXT)	ROL (EXT)	DEC (EXT)	•	INC (EXT)	TST (EXT)	JMP (EXT)	CLR (EXT)
8	SUB (A) (IMM)	CMP (A) (IMM)	SBC (A) (IMM)	•	AND (A) (IMM)	BIT (A) (IMM)	LDA (A) (IMM)	•	EOR (A) (IMM)	ADC (A) (IMM)	ORA (A) (IMM)	ADD (A) (IMM)	CPX (A) (IMM)	BSR (REL)	LDS (IMM)	•
9	SUB (A) (DIR)	CMP (A) (DIR)	SBC (A) (DIR)	•	AND (A) (DIR)	BIT (A) (DIR)	LDA (A) (DIR)	STA (A) (DIR)	EOR (A) (DIR)	ADC (A) (DIR)	ORA (A) (DIR)	ADD (A) (DIR)	CPX (A) (DIR)	•	LDS (DIR)	STS (DIR)
A	SUB (A) (IND)	CMP (A) (IND)	SBC (A) (IND)	•	AND (A) (IND)	BIT (A) (IND)	LDA (A) (IND)	STA (A) (IND)	EOR (A) (IND)	ADC (A) (IND)	ORA (A) (IND)	ADD (A) (IND)	CPX (A) (IND)	JSR (IND)	LDS (IND)	STS (IND)
B	SUB (A) (EXT)	CMP (A) (EXT)	SBC (A) (EXT)	•	AND (A) (EXT)	BIT (A) (EXT)	LDA (A) (EXT)	STA (A) (EXT)	EOR (A) (EXT)	ADC (A) (EXT)	ORA (A) (EXT)	ADD (A) (EXT)	CPX (A) (EXT)	JSR (EXT)	LDS (EXT)	STS (EXT)
C	SUB (B) (IMM)	CMP (B) (IMM)	SBC (B) (IMM)	•	AND (B) (IMM)	BIT (B) (IMM)	LDA (B) (IMM)	•	EOR (B) (IMM)	ADC (B) (IMM)	ORA (B) (IMM)	ADD (B) (IMM)	•	•	LDX (IMM)	•
D	SUB (B) (DIR)	CMP (B) (DIR)	SBC (B) (DIR)	•	AND (B) (DIR)	BIT (B) (DIR)	LDA (B) (DIR)	STA (B) (DIR)	EOR (B) (DIR)	ADC (B) (DIR)	ORA (B) (DIR)	ADD (B) (DIR)	•	•	LDX (B) (DIR)	STX (B) (DIR)
E	SUB (B) (IND)	CMP (B) (IND)	SBC (B) (IND)	•	AND (B) (IND)	BIT (B) (IND)	LDA (B) (IND)	STA (B) (IND)	EOR (B) (IND)	ADC (B) (IND)	ORA (B) (IND)	ADD (B) (IND)	•	•	LDX (IND)	STX (IND)
F	SUB (B) (EXT)	CMP (B) (EXT)	SBC (B) (EXT)	•	AND (B) (EXT)	BIT (B) (EXT)	LDA (B) (EXT)	STA (B) (EXT)	EOR (B) (EXT)	ADC (B) (EXT)	ORA (B) (EXT)	ADD (B) (EXT)	•	•	LDX (EXT)	STX (EXT)

DIR=ダイレクトアドレッシング
EXT=エクステンディットアドレッシング
IMM=イミディエイトアドレッシング

IND=インデックスアドレッシング
IMP=インプライドアドレッシング
REL=リラティブアドレッシング

A=Accumulator A
B=Accumulator B

5.メモリー容量の拡張について

ベーシックマスタージュニアＭＢ－６８８５はプログラムの作成上必要となるメモリー(ＲＡＭ：Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ　と呼ぶＩＣメモリーです。）の容量を拡張することができます。長いプログラムを作成する場合等で，標準実装の16Kバイトのメモリー容量では不足する場合は，メモリー容量を６４Ｋバイトにまで拡張することができますので，日立ベーシックマスター取扱店にご相談ください。有償にて処置させていただきます。なお，ご不明の点は同梱包されております「日立家電品ご相談窓口一覧表」をご参照のうえ，お近くの窓口にお問い合せください。（拡張の際の価格は前述取扱店にてご確認ください。)

ご注意

メモリー容量の拡張は全て，上記取扱店に用意しております日立の指定部品にて行います。セットをご持参いただきましたうえで拡張させていただき，所定の動作確認等を行いましたうえで，セットをお渡しいたします。（指定部品のみの販売はいたしません。)
万一，上記以外の方法により拡張を行なわれました場合は，保証期間内といえども保証をいたしかねる場合がありますのでご了承ください。

6．故障かな…と考える前に次のことをご確認ください

<table>
<tr><th></th><th>症　　　　　状</th><th>対　　　　　　　　策</th></tr>
<tr><td>1</td><td>電源スイッチが入らない。</td><td>電源コードの接続は大丈夫ですか（2頁参照）</td></tr>
<tr><td>2</td><td>音が出ない。</td><td>後面の音量ツマミが回してありますか</td></tr>
<tr><td>3</td><td>画面に無意味な記号や文字がでる。（特に電源スイッチを入れたとき。）</td><td>BREAK/RESET キー，カナ/記号 キーをおしてＲＥＳＥＴをかけてください。画面が正常になるはずです。（3頁参照）
（ＲＥＳＥＴで画面が正常になれば，セットは故障ではありません。）</td></tr>
<tr><td>4</td><td>ＲＵＮさせたところ，画面に何も出ないままになった。又は，何かが出るがいつまでたっても止まらなくなった。</td><td>・ＢＲＥＡＫをかけて実行を止め，プログラムにおかしい所がないか再度チェックしてみてください。
・ＢＲＥＡＫがかからないときは，プログラムにおかしい所があり暴走していることも考えられます。3頁の要領でＲＥＳＥＴをかけてみてください。</td></tr>
<tr><td colspan="3">何か正体不明の事態が生じた場合の最後の切札はＲＥＳＥＴをかけることですが，ＲＥＳＥＴをかけると，それまで作ったプログラムが全て消めつしてしまいます。
あまりＲＥＳＥＴのお世話にならないようにＲＵＮの前にプログラムにおかしい所はないか，充分チェックしてください。</td></tr>
<tr><td>5</td><td>テープのＳＡＶＥ，ＬＯＡＤ，ＶＥＲＩＦＹがうまくいかない。</td><td>97ページ以降の注意事項をもう一度ご確認ください。</td></tr>
<tr><td>6</td><td>ＭＵＳＩＣで音楽の演奏中ＢＲＥＡＫがうまくかからない。</td><td>音符の長さが長めのとき，特にＢＲＥＡＫがかかりにくくなります。ＢＲＥＡＫは1つの音符から，次の音符へのかわりめでかかるようになっていますので，BREAK/RESET キーを音符のかわりめまでおし続けてください。</td></tr>
</table>

付録

この項はベーシックマスタージュニアで，ＢＡＳＩＣのプログラミング，ならびにその実行を行なう上では特には必要としませんが，マイコンに精通された方がご利用いただくうえで何かの参考となればと思い掲載しましたのでご利用ください。

1. モニターのワークエリアとその内容

番地	バイト数			ラベル名	内容	初期設定値
	1	2	3			
0 0				USRNMI	ユーザーＮＭＩベクトル	(ＲＴＩ) Ｆ０Ｃ６
0 2				USRIRQ	ユーザーＩＲＱベクトル	(ＲＴＩ) Ｆ０６Ｅ
0 4				BREAKV	ＢＲＥＡＫキーからのＮＭＩ用ベクトル	B:BB7E M:F0D0
0 6				TIMERV	タイマー用割り込みベクトル	(ＲＴＩ) Ｆ０６Ｅ
0 8				RAMEND	ＲＡＭの最大番地	16Kバイト ３ＦＦＦ
0 A				TIME 1	上位バイト　2^{16}秒≒18時間ソフトウェアタイマー	0000
0 B				TIME 2	下位バイト　0≦TIME 2≦$FF	
0 C				TIME 3	1秒タイマー　0≦TIME 3≦59	0 0
2 8				ASCIN	1文字入力用ジャンプテーブル	7EFA55
2 B				ASCOUT	1文字出力用ジャンプテーブル	7EF7AB
2 E				BYTIN	メモリーブロック入力用ジャンプテーブル	7EF71F
3 1				BYTOUT	メモリーブロック出力用ジャンプテーブル	7EF619
3 B				RECTOP LDTOP MSTTOP	カセット記録 カセット再生 メモリー転送　の先頭番地	
3 D				RECEND LDMAX MSTEND	カセット記録 カセット再生 メモリー転送　の最終番地	
3 F				CPYTOP	メモリー転送のコピー先の先頭番地	
4 1				CPYEND	メモリー転送のコピー先の最終番地	
4 3	8バイト			FNAME	カセット記録，再生時のファイル名	

上記の表のほか，モニターのワークエリアとして＄００～＄７１番地を，ＢＡＳＩＣのワークエリアとして，＄７２～＄ＦＦおよび，＄４００～＄９ＦＦを使用しています。ＢＡＳＩＣのユーザープログラムは＄Ａ００番地から格納されます。

上記の表に記されたワークエリアは，システムを拡張して外部にプリンターなどのインターフェースを付ける場合などに，次頁のモニターのサブルーチンのワークエリアとなっている部分です。

2. モニターのサブルーチン概要

実行後のレジスターの状態：◎；必要な情報へ変化，○；実行前と変化なし，—；不特定な値に変化（破壊）

スタートアドレス	ラベル名	レジスターの状態			内容
		IX	A	B	
００２８ レジスター (００２９ ００２Ａ)	ASCIN	○	◎	○	・＄２９，２Ａ番地で指定されるアドレスの入力サブルーチンによりＡＣＣＡに１文字入力する。 通常はＣＨＲＧＥＴが接続され，キーボードからＪＩＳコードを１文字入力します。
００２Ｂ レジスター (００２Ｃ ００２Ｄ)	ASCOUT	○	○	○	・＄２Ｃ，２Ｄ番地で指定されたアドレスの出力サブルーチンにより，ＡＣＣＡの１文字を出力する。 通常はＣＨＲＯＵＴが接続され，ＣＲＴディスプレイに１文字出力します。
００２Ｅ レジスター (００３Ｂ ００３Ｄ ００４３)	BYTIN LDTOP LDMAX FNAME	—	—	—	・指定装置からデータブロックを入力する。 左記レジスターのＦＮＡＭＥで指定されたデータブロックを，レジスターに示されたＬＤＴＯＰからＬＤＭＡＸのアドレスに格納します。相対番地入力のときは，ＬＤＴＯＰ以下の番地に順次入力される。通常はＬＯＡＤに接続されている。
００３１ レジスター (００３Ｂ ００３Ｄ ００４３)	BYTOUT RECTOP RECEND FNAME	—	—	—	・指定装置にデータブロックを出力する。 左記レジスターのＦＮＡＭＥで定義された，ＲＥＣＴＯＰから，ＲＥＣＥＮＤまでのデータブロックを出力します。通常は，ＳＡＶＥに接続されている。
Ｆ００３	ADDIXB	◎	○	○	・(ＩＸ)←(ＩＸ)＋(ＡＣＣＢ) インデックスレジスターにＡＣＣＢの内容を加算する。
Ｆ００９ レジスター (００３Ｂ ００３Ｄ ００３Ｆ)	MOVBLK MSTTOP MSTEND CPYTOP	—	○	○	・データブロックを転送する。 左記レジスターのＭＳＴＴＯＰから，ＭＳＴＥＮＤのアドレスのデータをＣＰＹＴＯＰ以下のアドレスに転送する。コピー先の最終番地は計算されて＄００４１，２番地に書き込まれる。モニターのＴコマンド参照。
Ｆ００Ｆ	KBIN	○	◎	○	・キーボードの状態を読みとり，ＡＣＣＡに文字，図形コードを入れる。キーボードがおされているとき，コンディションコードレジスターのCbitが0になる。 例　KYBD　JSR　KBIN　BDF00F 　　　　　BCS　KYBD　25FB 　　　　　JSR　$002B　BD002B 　　　　　BRA　KYBD　20F6

実行後のレジスターの状態：◎；必要な情報へ変化，○；実行前と変化なし，—；不特定な値に変化（破壊）

スタートアドレス	ラベル名	レジスターの状態 IX	A	B	内容
F012	CHRGET	○	◎	○	・キーボードから1文字入力する。 ・カーソルとしてACCAの内容を出力する。 $F00F番地のKBINとの違いは，前頁の例の1行めを下記のようにすればわかります。 JSR　$F012　BDF012
F015	CHROUT	○	○	○	・CRTディスプレイに1文字出力する。 ・$01～$0Fはコントロールコードで，RETURN時CARRY＝1となる。
F018 レジスター (003B 003D 0043)	LOAD LDTOP LDMAX FNAME	—	—	—	・カセットテープからデータブロックを入力する。 左記レジスターのFNAMEで指定されたテープのデータをメモリーに格納する。格納したいアドレスの先頭番地はレジスターのLDTOPに入れてサブルーチンをコールすると最終番地はLDMAXに記録される。
F01B レジスター (003B 003D 0043)	SAVE RECTOP RECEND FNAME	—	—	—	・データブロックをカセットテープに出力する。 左記レジスターのFNAMEで定義された，レジスターのRECTOPからRECENDまでのデータブロックをカセットテープに出力します。
FFE6	NMISET	○	—	○	・BREAK割り込みを禁止する。 JSR　$FFE6を実行するとBREAK割込みが禁止される。
FFE9	NMICLR	○	—	○	・BREAK割り込み禁止を解除する。 JSR　$FFE9を実行すると，禁止されていたBREAK割り込みが許可される。
FFEC	OUTIX	○	—	○	・カーソル位置にIXの内容を16進表現で表示する。
FFEF	CLRTV	—	—	—	・画面を消去し，カーソルをホームポジションに移動する。
FFF2	MESOUT	—	—	○	・IXで示された番地以下のJISコードを$04がくるまで出力を表示する。
FFF5	CURPOS	◎	○	○	・カーソル位置の絶対番地をIXに入れる。

3．高速カセット入出力命令の使い方

ベーシックマスタージュニアは，カセットテープレコーダへのプログラムの記録・再生を取扱説明書3.11プログラムの記録・再生に記載の方法に比べ4倍の速さで行なえる機能を内蔵しております。

ご使用に当りましては，以下の説明をよくお読みいただき，正しくご使用ください。

(1) 特　長

①高速データ入出力

高速カセット入出力命令（本付録(3)カセット入出力命令参照）を使用することにより，ＢＡＳＩＣおよび機械語プログラムの記録および再生格納速度は，取扱説明書3.11プログラムの記録・再生に記載の方法に比べ4倍（３００ボー→１２００ボー）となり，プログラムの記録・再生時間を大幅に短縮できます。

〈ご注意〉

ボー：データ転送速度を示す単位。３００ボーは約２７文字／秒，１２００ボーは約１０９文字／秒の転送速度です。

②サウンドモニター

高速カセット入出力命令によるデータ入出力時，連続音を発生し記録・再生状態の動作確認が可能です。なお，音量調節は本体背面にある音量調節ツマミを調節してください。

③プログラムオートスタート

高速カセット入出力命令を使用することにより，プログラムオートスタートが可能です。これは，プログラムをカセットテープから再生格納後ただちにそのプログラムを実行する機能です。

ＢＡＳＩＣプログラムの場合，先頭の行番号より，機械語プログラムの場合，プログラム記録時の先頭番地より実行します。

④ＢＡＳＩＣプログラム中でＳＡＶＥ，ＬＯＡＤ可能

高速カセット入出力命令を使用することにより，ＢＡＳＩＣプログラムの中で他のＢＡＳＩＣおよび機械語プログラムを記録したり，再生格納したりすることが可能です。

(2) 使用上のご注意

① 取扱説明書3.8データ入出力に関するステートメント，および3.11プログラムの記録・再生に記載の方法を使用した場合，データ転送速度は３００ボーです。

また，本付録(3)カセット入出力命令に記載の方法を使用した場合，データ転送速度は１２００ボーとなります。

② カセットテープレコーダ（ＴＲＱ－２３７およびＴＲＱ－３５９）の再生レベル目盛りは4から6の範囲で，またＴＲＱ－３５９の音質レベル目盛りは3から5の範囲でご使用ください。

③ 記録時に使用したカセットテープレコーダーと異なるものを再生時に使用した場合，正常に再生できないことがありますので，記録時と再生時はなるべく同一のカセットテープレコーダーをご使用ください。

④ 本セットのカセット記録再生特性は，日立カセットテープレコーダーＴＲＱ－２３７およびＴＲＱ－３５９を基本に設定してあります。他のカセットテープレコーダーを使用した場合，録音・再生特性が異なるために正常に再生されないことがあります。このような場合には，取扱説明書3.11プログラムの記録・再生記載の方法で記録・再生を行ってください。

(3) カセット入出力命令

<ご注意>

1．本命令に続くマルチステートメントは許されません。
2．本命令の実行中は，タイマーのカウント機能が動作しません。
3．ＢＡＳＩＣプログラムはＲＡＭ上に記憶されている形式(内部表現方式)でカセットテープに記録します。

①ＳＡＶＥ（セーブ）

機能：ＢＡＳＩＣおよび機械語プログラムの記録

書式：(a) ＣＡＬＬ＄Ｅ４００：ＳＡＶＥ <ファイル名>

(b) ＣＡＬＬ＄Ｅ４００：ＳＡＶＥ <ファイル名>，<先頭番地>，<最終番地>

解説：(a) ＢＡＳＩＣプログラムをカセットテープに記録します。

(b) 機械語プログラムをカセットテープに記録します。

<ファイル名>は，“，”(コンマ) を除く最大６文字までの英大文字，英記号，カナ文字，カナ記号および数字が使用できます。<ファイル名>を省略した場合は，無名ファイル(ファイル名なし)として記録されます。機械語プログラムの <先頭番地>および<最終番地>は，“＄”から始まる１～４桁の１６進数(０～９，Ａ～Ｆ)で指定します。

ＳＡＶＥ命令をダイレクト実行で行うときは，テープへの記録が始まるとファイル名とプログラムの形式（Ｓ：ＢＡＳＩＣプログラム，Ｂ：機械語プログラム）を表示します。

ＢＡＳＩＣプログラム中で本命令を実行するときには，これらを表示しません。

例1 ＣＡＬＬ＄Ｅ４００：ＳＡＶＥテスト＃０１

例2 １０ＣＡＬＬ＄Ｅ４００：ＳＡＶＥテスト＃０２，＄２０００，＄２ＦＦＦ

<ご注意>

1．７文字以上のファイル名を指定するとエラー（ＥＲＲＯＲ３）となります。
2．機械語プログラムを記録するときには，必ず<先頭番地>と<最終番地>を指定してください。この指定のないときは，メモリー上のＢＡＳＩＣプログラムが記録されます。

②ＶＥＲＩＦＹ（ベリファイ）

機能：ＳＡＶＥ命令でカセットテープに記録したプログラムと，メモリー上のプログラムとの比較

書式：ＣＡＬＬ＄Ｅ４００：ＶＥＲＩＦＹ<ファイル名>

解論：ＳＡＶＥ命令でカセットテープに記録したＢＡＳＩＣまたは機械語プログラムとメモリー上のプログラムとの比較確認を実行します。但し<ファイル名>を省略したときには，最初に検出したプログラムとメモリ上のプログラムとの比較確認を行います。

ＶＥＲＩＦＹ命令をダイレクト実行で行うとファイル名とプログラムの形式を表示し，比較確認が終了すると，次の行にプロンプト記号(＞)を表示します。

例3 ＣＡＬＬ＄Ｅ４００：ＶＥＲＩＦＹテスト＃０１

③ＬＯＡＤ（ロード）

機能：ＢＡＳＩＣおよび機械語プログラムの再生格納

書式：(a) ＣＡＬＬ＄Ｅ４００：ＬＯＡＤ<ファイル名>

(b) ＣＡＬＬ＄Ｅ４００：ＬＯＡＤ<ファイル名>，Ｒ

解説：(a) カセットテープ上のＢＡＳＩＣまたは機械語プログラムの再生格納を行います。

(b) カセットテープ上のＢＡＳＩＣまたは機械語プログラムを再生格納後，ただちにそのプログラムを実行します。(これをオートスタート機能と称します。)

〈ファイル名〉を省略したときには，最初に検出したプログラムを再生格納します。ＬＯＡＤ命令をダイレクト実行で行うときは，ファイル名を検出するとファイル名とプログラムの形式を表示し，再生格納が終了すると次の行にプロンプト記号を表示します。ＢＡＳＩＣプログラム中で本命令を実行するときには，これらを表示しません。，Ｒを追加した場合は，ＢＡＳＩＣプログラムの場合，先頭の行番号より，機械語プログラムの場合，プログラム記録時の先頭番地より実行します。

例4　ＣＡＬＬ＄Ｅ４００：ＬＯＡＤテスト＃０１

例5　１０ ＣＡＬＬ＄Ｅ４００：ＬＯＡＤ，Ｒ……ＢＡＳＩＣプログラムにおいて別のプログラムを再生格納終了後ただちに実行する。

〈ご注意〉

1．ＢＡＳＩＣプログラムの再生格納を開始すると，ＲＡＭにあるＢＡＳＩＣプログラムを消去した後に新しいプログラムを格納します。ただし，指定したプログラムを検出するまではＲＡＭ上のプログラムは保護されます。
2．ＢＡＳＩＣプログラムにおいて別のＢＡＳＩＣプログラムを再生格納する場合，，Ｒを指定しないと再生格納終了後そのプログラムは終了し，コマンド待ち状態に戻ります。このとき，もとのプログラムは破壊されますが，再生格納したプログラムは有効です。
3．ＢＡＳＩＣプログラムと機械語プログラムを同一のファイル名で記録した場合，先に記録してあるプログラムが再生格納されます。
4．機械語プログラムはプログラム記録時と同一番地に再生格納されます。このときそのエリアに別のプログラムが存在しても常に新しいプログラムが優先して格納されます。

(4) エラーメッセージ

エラー表示	エラー内容
＊ＥＲＲＯＲ （ＶＥＲＩＦＹでのエラー）	・カセットに記録したプログラムとメモリー上のプログラムが一致しない。 ・プログラムが正しく再生できない。 この場合，次の項目を実施してみてください。 ①再びVERIFYを行う。再びSAVEした後VERIFYを行う。 ②テープレコーダーの出力レベル目盛りを可変してVERIFYを行う。 ③カセットテープを取替えてSAVE，VERIFYを行う。 ④トーンコントロール（音質調整）付きのテープレコーダーの場合，音質を変化させてVERIFYを行う。 ⑤テープレコーダーに異常がないか調べる。
＊ＥＲＲＯＲ （ＬＯＡＤでのエラー）	・プログラムが正しく再生できない。 この場合，次の項目を実施してみてください。 ①再びLOADを行う。再びSAVEした後LOADを行う。 ②テープレコーダーの出力レベル目盛りを可変してLOADを行う。 ③カセットテープを取替えてSAVE，LOADを行う。 ④テープレコーダーの音質を変化させてLOADを行う。 ⑤テープレコーダーに異常がないか調べる。
＊ＭＯＤＥ ＥＲＲＯＲ （ＬＯＡＤでのエラー）	・プログラム記録時とグラフィックページ使用状態が異る。 この場合，本付録＊MODE ERROR発生時の対処方法を参照してください。
ＥＲＲＯＲ ３	・ファイル名が6文字を超えている。

(5)　*MODE ERROR発生時の対処方法

本カセット入出力命令ではＢＡＳＩＣプログラムの記録・再生の高速化を可能とするためにＢＡＳＩＣプログラムをＲＡＭ上に記憶されている形式(内部表現形式)でカセットテープに記録する方法を採用しています。このため，プログラム記録時と再生格納時とのグラフィックページ使用状態が異ると＊ＭＯＤＥ　ＥＲＲＯＲが発生し，プログラムは再生格納できません。この場合，以下に示す方法で再度再生格納あるいは記録を行ってください。

①方法１〈プログラム記録時と同一のグラフィックページを設定する場合〉

取扱説明書3.13高解像度グラフィックの使い方に記載の命令で，プログラム記録時と同一になるようグラフィックページを再設定の後，再度再生格納を行う。

②方法２〈プログラムを目的のグラフィックページに適合させる場合〉

以下に示す方法により，目的のグラフィックページに適合したプログラムをカセットテープに再記録することができます。

(a) 方法１に示す方法でプログラムを再生格納する。

(b) 取扱説明書3.11プログラムの記録・再生に記載の命令によりプログラムを記録する。

　　ＳＡＶＥ〈ファイル名〉

(c) 目的のグラフィックページに再設定する。

(d) 取扱説明書3.11プログラムの記録・再生に記載の命令により，(b)で記録したプログラムを再生格納する。

　　ＬＯＡＤ〈ファイル名〉

(e) 本付録(2)カセット入出力命令に示す命令により，プログラムを記録する。

　　ＣＡＬＬ＄Ｅ４００：ＳＡＶＥ〈ファイル名〉

仕様（本仕様は改良のため，予告なく変更することがあります。）

■形式名	MB-6885
■プログラミング言語	BASIC，機械語及びアセンブラ（オプション）
■MPU	6800
■ROM	18Kバイト
■RAM	16Kバイト標準実装
■表　示	文字表示構成 横32字×縦24行（768字） 表示内容 文字およびグラフィック記号253種 グラフィック表示構成 横256ドット×縦192ドット}の2モード（ソフトウェア切替） 横 64ドット×縦 48ドット}
■画面コントロール	自動スクローリング，白黒反転可（プログラム切換） グラフィックマルチページ2頁（プログラム切換）
■キーボード	JIS配列準拠 56キー
■内蔵インターフェイス	カセットテープインターフェイス 300ボー／1200ボー カンサスシティスタンダード モノクロビデオインターフェイス 複合映像信号出力 分離映像信号出力 プリンターインターフェイス セントロニクス仕様準拠パラレルインターフェイス
■拡張インターフェイス	1ポート
■電源電圧	AC100V±10%　50／60Hz
■消費電力	13W（カラーアダプター，メモリー等を拡張しない標準状態）
■使用環境条件	温度0℃～35℃ 湿度20%～80%　但し結露のないこと。
■保存温度	－15℃～60℃
■外形寸法	39.5（幅）×9.7（高さ）×32.7（奥行）cm
■重　量	4.5 kg

保証とアフターサービスについて

■保　証

保証期間中，万一，故障した場合には，保証書記載事項に基づき無償修理いたしますので，お買い求めの販売店に保証書をご提示ください。なお，保証期間中でも有償となることがありますので，保証書をよくお読みください。

■転居後のアフターサービス

ご転居により，お買い求めの販売店のアフターサービスを受けられなくなる場合は，前もってお買い求めの販売店にご相談ください。ご転居先での日立の家電品販売店を紹介させていただきます。

■補修用性能部品の保有期間

ベーシックマスターの補修用性能部品の最低保有期間は８年間です。

詳しくはお買い求めの販売店（または当社のサービス窓口）にご相談ください。

■このベーシックマスタージュニアを使用できるのは日本国内のみで，外国では使用できません。

This PERSONAL COMPUTER set can be used only in Japan.

日立家電販賣株式會社

〒105 東京都港区西新橋2丁目15番12号
電話(03)502－2111

株式會社 日立製作所

〒105 東京都港区西新橋2丁目15番12号
電話(03)502－2111

…04

Printed in Japan YM-S (H)